BGPMon: the Next Generation

Spiros Thanasoulas
Christos Papadopoulos
Colorado State University
April 1, 2015

Work supported by NSF and DHS through a grant from the Australian government
The Team

Dan Rammer
Spiros Thanasoulas
Ela Sienkiewicz
Tyler Scott
Kaustubh Gadjkari
Christos Papadopoulos
Alison Craig
Darshan Wash
Anant Shah
BGPmon: the Past

- Service model:
  - XML streaming - updates and RIBS
  - Perl toolkit to manipulate XML
- Scaling based on chaining
- Version 7.4 (Jan 2015) was the last release based on the old mode to include new features
- Version 7.5 will include bug fixes only
- v8 will be a complete rewrite
New in v7.4

- Bug fixes (many more left)
- Geolocation of peers and collectors, down to city level
  - 330 peers, IPv4 and IPv6
  - configuration file, read on startup
  - local operators can add more peers as needed
- Offline MRT2XML translator
- Improved logging
- OpenBSD support
The New BGPmon Node Architecture
Cassandra Cluster
Private BGPmon Deployment

- Networks can deploy independent instances of BGPmon
- Interconnection options:
  - None
  - Import only
  - Import/Export
Tracking Outages (in progress)

• Outage data provided by the LACREND project at ISI (PI: John Heidemann)
• 2nd order information, derived from ISI’s ping sweeps
• Two community services (in progress):
  – Public outage DB
  – Outage visualization with annotation capabilities
OutageDB

- OutageDB contains:
  - Outage information
  - BGP messages before and after the outage
- Research question: Can we model outages at the control plane and predict an outage is about to happen?

<table>
<thead>
<tr>
<th>Outage ID</th>
<th>IPBlock</th>
<th>BGP_LPM</th>
<th>BlockAGgr</th>
<th>Outage Start</th>
<th>Outage End</th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>128.125.92.0/24</td>
<td>128.125.0.0/16</td>
<td>128.125.0.0/17</td>
<td>2012-10-27 06:20:21</td>
<td>2012-11-08 18:16:03</td>
</tr>
</tbody>
</table>

Substantial update traffic brackets outages
Outage Visualization and Annotation

• OutageDB client
• Filters:
  – Time range
  – Country
  – Prefix
• Google login for annotations (TBD)
BGP Stats (in progress)

- Real time statistics from the BGPmon archive
- Client of the BGPmon archive
- Filters:
  - All BGP
  - NLRI
  - MP Reach
  - MP Unreach
  - Withdraw
  - Per peer and location (TBD)
BGPMon in a Nutshell

• collect BGP messages
• store them
• serve the user queries:
  – return sets of stored messages
  – reduce sets of stored messages to results
  – transform messages to other formats
General Requirements

• Lightweight core
  – hundreds of BGP peers. thousands of client queries

• Extensible
  – support different input/output formats

• Scalable
  – utilize all available cores
  – Easy node addition

• Language-agnostic Ecosystem

• Leverage available technologies, industry proven best practices
Requirements - Core

• Minimal BGP talker
  – everything in / keepalive out

• HTTP as client interface
  – directly modeling the request/response paradigm
  – easily cacheable

• The core is a request/response router
Requirements - Scalability and Data Interchange

• Concurrent message passing
• Lock-less due to CSP-style synchronization
• Robust Storage
• Flexible wire format
Technologies

• Language: Golang
  – mature (7 years old)
  – compiled and static typed
  – concurrency primitives (typed channels/goroutines)
  – rich standard library (net/, bufio/, encoding/{json, xml})
  – multiplatform (Linux,{Free, Open, Net}BSD, OSX, Windows for x86/arm/amd64)

• Storage: Cassandra
  – elastic scalability
  – transaction support
    • atomicity
    • tunable consistency
  – automatic cluster synchronization

• Internal Format: protocol buffers
  – compilers for most languages
  – heavily optimized for efficient transfer over the wire
  – automatic JSON decoration for debugging
BGPmon Robustness
Example: Replication Factor 2
One Link Loss – System is Robust
Two Link Loss – AU Data Saved Locally
Connectivity Restored – Data Added to Cassandra Cluster
So where are we now?

<table>
<thead>
<tr>
<th>collect</th>
<th>store</th>
<th>return</th>
<th>reduce</th>
<th>transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>80%</td>
<td>25%</td>
<td>90%</td>
<td>50%</td>
<td>20%</td>
</tr>
</tbody>
</table>

• Fully fledged concurrent bgpd implementation
  – Mainly developed by OSRG, NTT, Japan
  – JSON configuration and control over HTTP

• Our additions to it include:
  – gomrt
    • pure golang library to read and write MRT data
    • builds on the bufio interfaces
  – RPC control plane
Current Archive

• HTTP interface to expose the data in a unified format that abstracts away file structure

• Prototype for the RESTful BGPMon interface
  – specify date range and the format
  – gomrt to parse deeper in MRT
  – Return result over an HTTP 1.1 chunked encoding channel

• Allows caching of most popular data

• Handles thousands of concurrent requests on pretty much commodity hardware
The New User Interface

• Simple HTTP-based **pull model**
  – NOT the same as pulling files from the RouteViews archive
• Request parameters:
  – Time range (s) - **YYYYMMDDHHMMSS**
  – Data type (updates or RIBs)
  – Data format (currently MRT - TBD: XML, JSON)
  – Statistics (TBD)
  – Connectivity maps (TBD)
• Example request (try it!):
  – `$> curl -o outputfile http://bgpmon.io/archive/mrt/updates?start=20150301000000\&end=20150301010000`
  – `$> curl -o outputfile http://bgpmon.io/archive/mrt/ribs?start=20150301000000\&end=20150301010000`
Cassandra - RV Integration

• Currently experimenting with appropriate schemas to store our BGP data
• Cassandra allows building a schema with the indexing variables living next to the whole protobuf blob
• Over 10 years of RouteViews data (11TB as of 2015)
  – We are an official mirror
• Archive will be imported into the Cassandra and made available through the new BGPmon
stats

• The stats server is client to the archive
• Same request format, returns statistics in JSON:
  – withdrawals
  – NLRI
  – mpreach/mpunreach
• We provide client-side javascript to render graphs on the browser
testing/deployment

• Everything is dockerized
• We provide docker images that bundle Cassandra, golang, protobuf compilers and our gobgp code
• We plan to distribute BGPMon as a docker image
Future Work

• Add more information to the underlying DB
  – Outages
  – Data plane (ISI’s pinger project)
  – Hijacks
  – More?

• Cloud hosting?
BGPmon networks

• Data sharing between nodes in a monitoring network
  – Just configuring their Cassandras to be in the same cluster
    • automatic replication
  – CSU will maintain such an infrastructure with public data
    • users can choose between running their own BGPmons and having their Cassandras join an existing cluster
    • or just peer with a BGPmon and access their data over the RESTful interface
    • pushing data to the dbs has to be done by an approved user of our public cluster
    • if an organization so desires, they can run their own private cluster/instance
    • they will be still be able to pull all the public data we provide