BGP Parser Implementation

Jason Ryder, Alex Wang, Paul Wang
UCLA
CS 217B, Spring 2009
Overview

- Implementation of a BGP, MRT (Multi-threaded Routing Toolkit) and XFB (XML Format for BGP) parser
- MRT and XFB can be used to encapsulate BGP data
- These data are archived/maintained by RIPE and Routeviews.
- C++ Implementation for extensibility
Motivation

- Needed for research in the area of routing
- Existing tool (libbpgdump) has some deficiencies
  - Only handles withdrawals OR announcements from a BGP message (it is possible to have BOTH)
  - Lacking full support for multicast and 4 byte AS numbers
  - Some attributes are ignored
  - Not easily extensible
Initial Challenges

- No centralized specification
  - Specifications are spread out over numerous RFCs and other spec documents
  - *Conflicting* specifications – *we* found an error in the specification for MRT format
  - Some specifications were not crystal clear. Implicit assumptions existed in some cases.
    - Message formats within MRT were sometimes taken from BGP, but not always expressly stated
- Time constraint – very limited time to create fully functioning parser (esp. considering there are many *hundreds* of pages of specification)
Metrics

- Extensibility – BGP is ever-evolving
- Efficiency – Massive amounts of data
- Correctness – Handle cases that currently are not
- Robustness – Good error handling and reporting
Implementation

- C++ was used for low-level memory access and ease of future extensibility
- Object-oriented, polymorphic approach
- Factory pattern used for creating specific object types from a binary stream
- Extensive standard template library (STL)
- Support for multiple platforms (Linux, Windows, Mac)
Methodology

- Initial research required to familiarize ourselves with the protocol
- Version control (via Google projects)
  - Link: http://code.google.com/p/bgpparser-ucla/
- Agile development process
  - Running code at the end of each week
  - Regular meetings for status checks and accountability
- Extreme Programming practices
  - Iterative development
  - Pair development
Code Hierarchy – BGP Message
Code Hierarchy – Route

```
Route
#length : unsigned char
#prefix : IP Address
#numOctets : int
+getLength() : unsigned char
+getPrefix() : IP Address
+getNumOctets() : int
+printMe() : void
```

- NLRIReachable
- NLRIUnReachable
Demonstration
Testing/Debugging

- Manual binary verification
- Test runs on multiple types of data against libbogpdump
- Special case input (eg: withdraw+announce)
- Code review
- Pair programming
Conclusions

- Successfully parsed broad range of given messages
- Non-centralized and inconcise specifications cause implementation problems
- Theory and Practice are extremely different
- You must challenge even what may be considered “correct”
Questions?