From traditional to alternative approach to storage and analysis of flow data

Petr Velan, Martin Zadnik

Introduction

- Network flow monitoring
  – Visibility of network traffic

- Flow analysis and storage enables
  – Maintenance
  – Security
  – Planning

Introduction

- Traditional tools
  - Fixed format of flow data
  - Storage optimization
  - Offline analysis of 5 minutes intervals
- Latest trends
  - Increasing amount of flows
  - Network applications
  - Attacks
- Alternative approach

Motivation

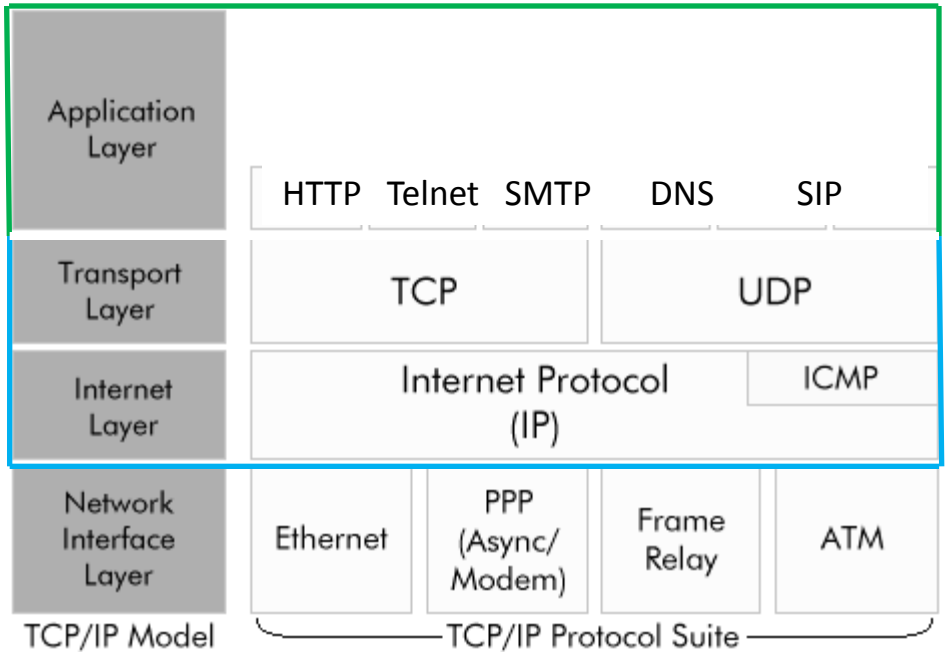
- Trend is to connect anything
  - Mobile devices
  - Internet of Things, Everything

SmartTV

SmartToilet

Motivation

- Migration of applications into clouds
  – Secure network environment supports trustworthy service delivery to end users
- Application-oriented attacks
  – Visibility of application layer

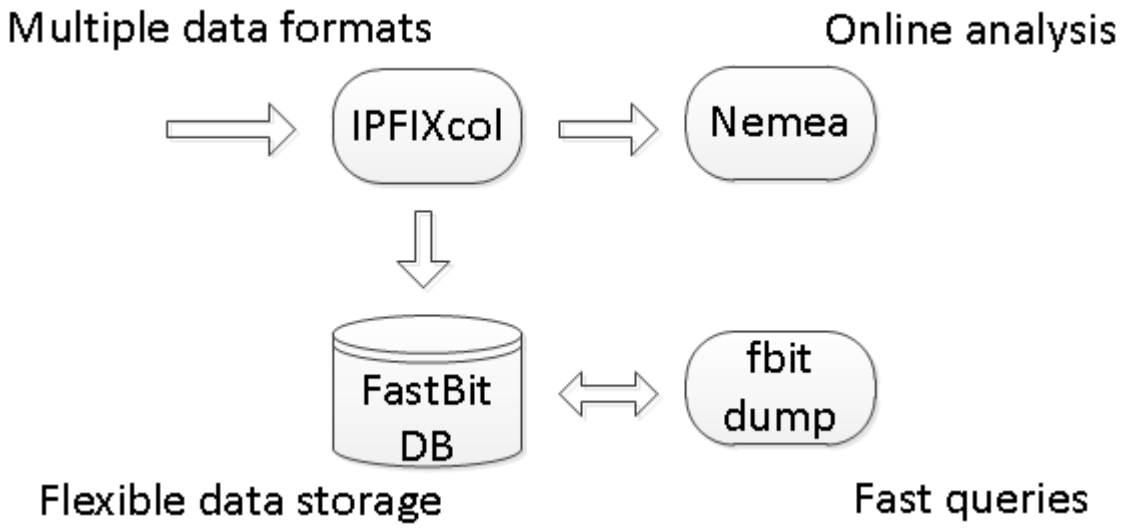


Issues

- Various flow data formats
- Flexibility of data storage
- Response to data queries
- Online analysis and effective data processing

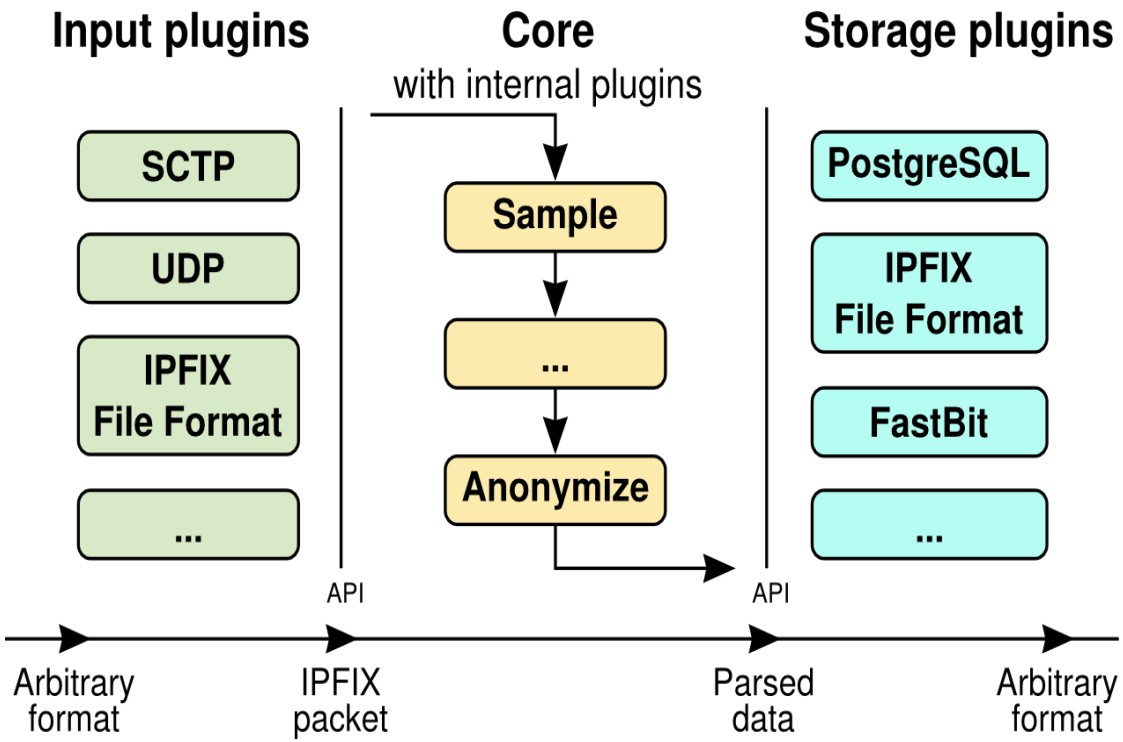
Concept

- Modular collector to support various formats
- Database to store dynamic data
- Column oriented storage for fast queries
- Streaming processing for online analysis



Collector

- IPFIXcol
  - IPFIX protocol support (RFC 7011)
  - Modular architecture



Collector

- IPFIX records from network or local file
  - Private Enterprise Numbers
  - Variable length elements
  - Easily extensible for new elements
- Internal record processing → IPFIX mediator capabilities (RFC 6183)
- Data output:
  - Local storage
  - Further processing
- Primary data storage: **FastBit** database
- Further data processing using **Nemea**

Streaming analysis

- Traditional concept is to store & analyze
  - 5 minutes interval
  - Intensive disk access
- Alternative streaming approach
  - Analysis pipeline of multiple modules
  - Stream data through modules
  - Keep data in memory
- Nemea framework

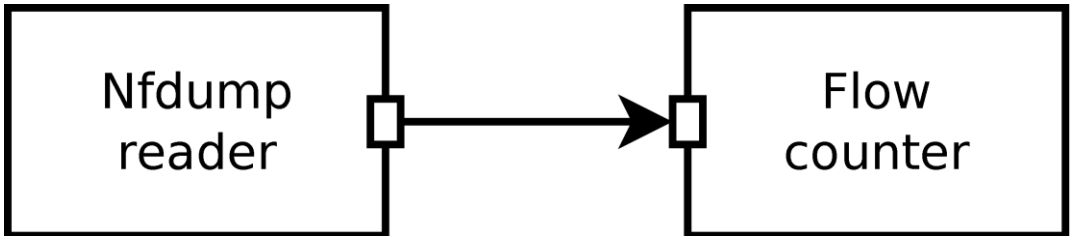
Streaming analysis

- Unified interface for efficient intermodule communication
  - Unix sockets (single host)
  - TCP sockets (multiple hosts)

- Data format defined at run time (binary)
- Efficient data transfer
- Blocking/non-blocking
- Connection recovery
- Supervisor

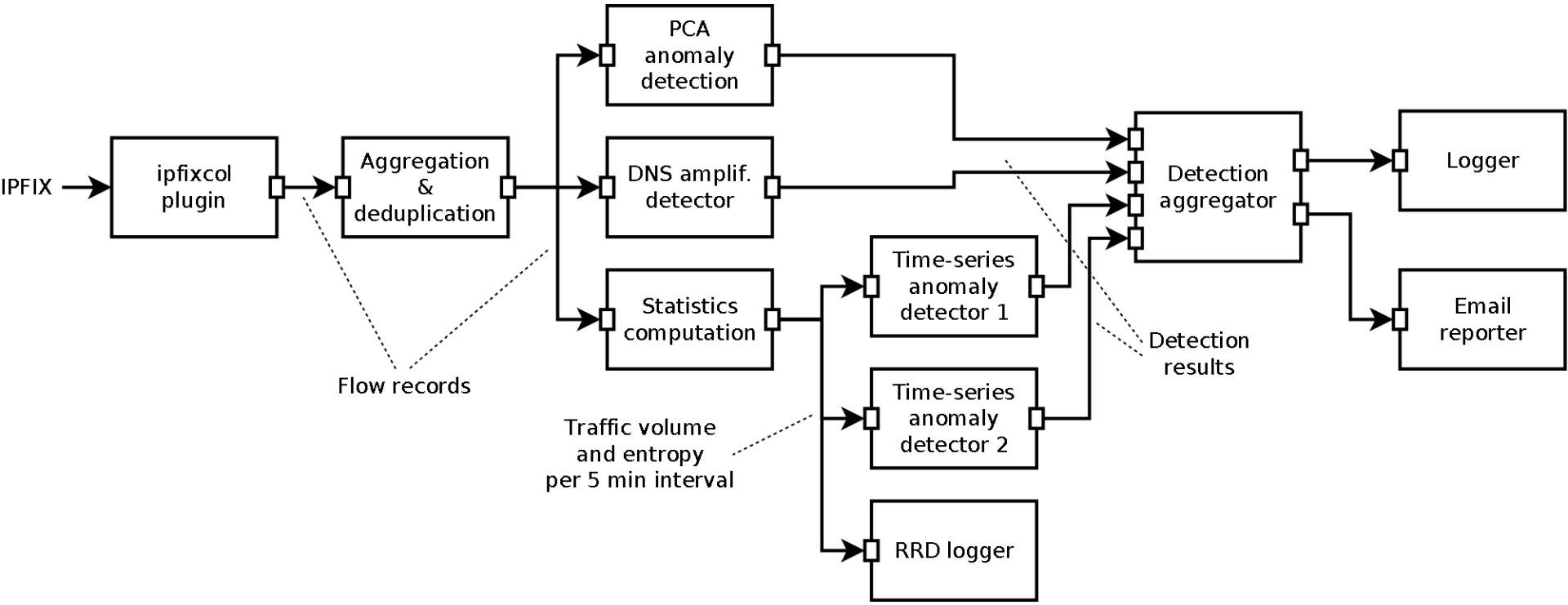
Streaming analysis

- Simple offline example
  - Stream stored flows
  - Count number of received flows



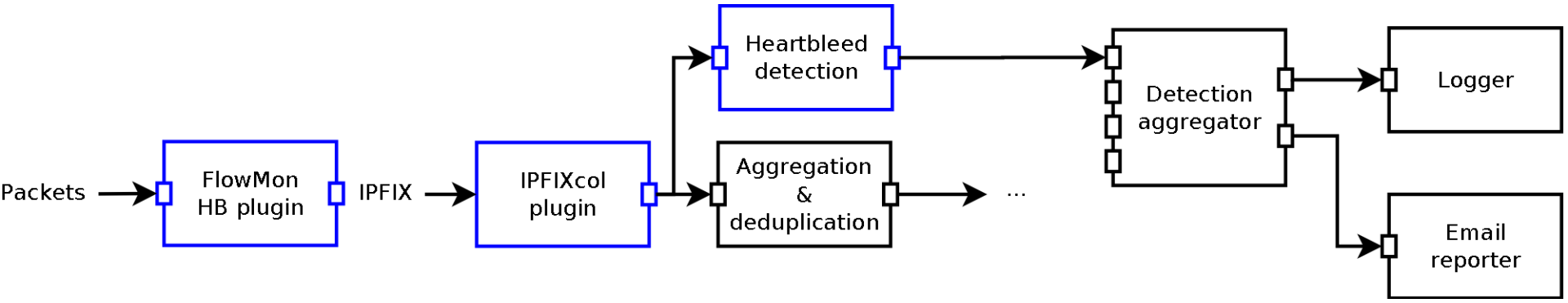
Streaming analysis

- Advanced online example
  - IPFIX stream → Nemea stream
  - Multicast stream to modules



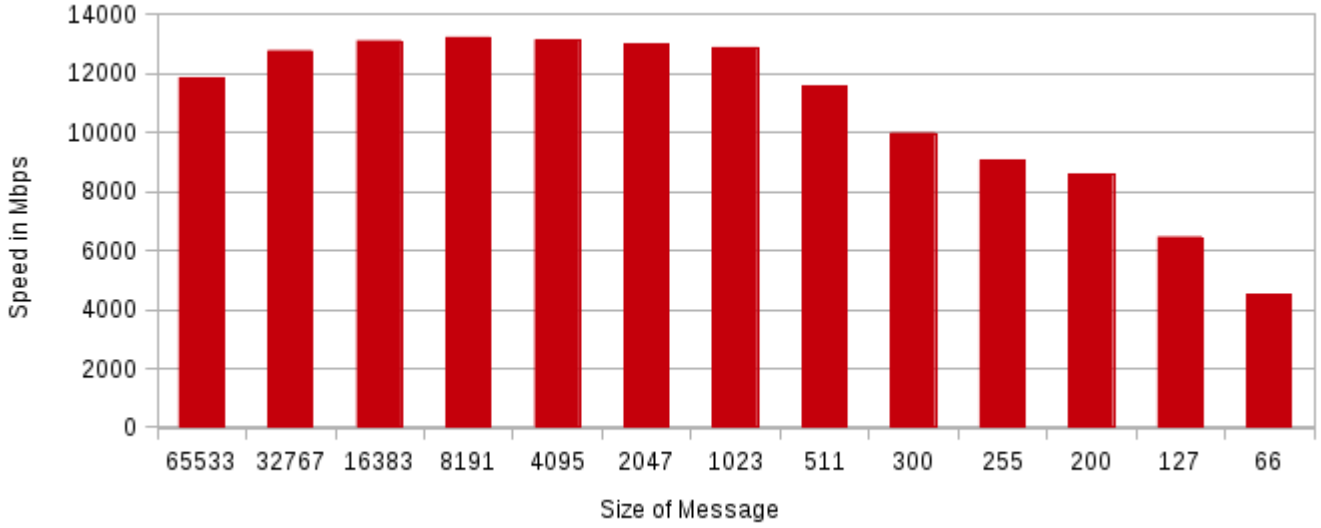
Streaming analysis

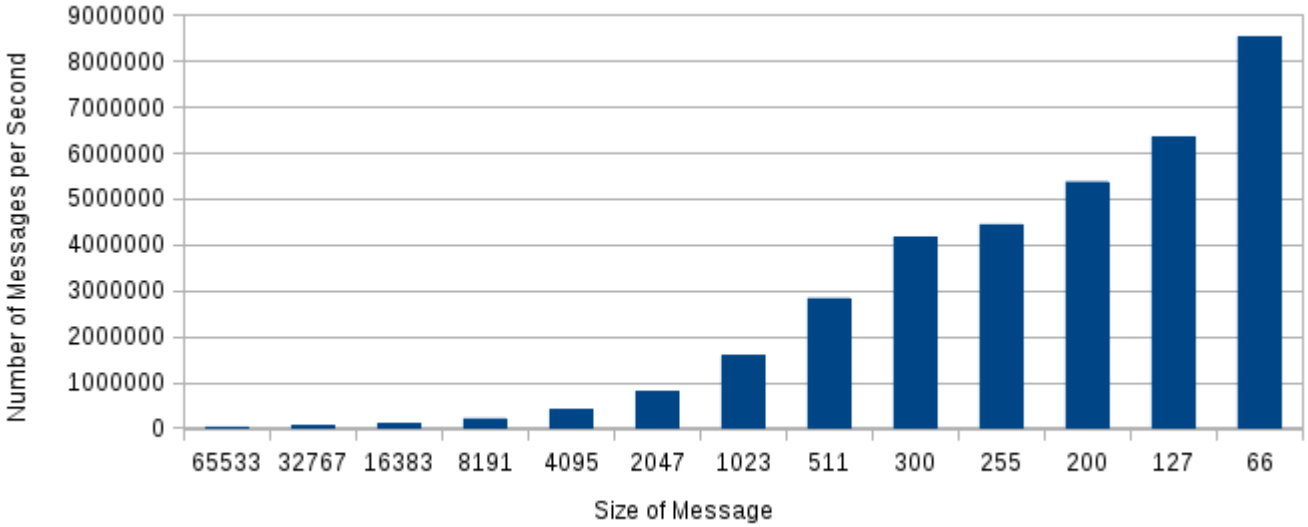
- Adding new monitoring/analysis capability
- Use case
  - Heartbleed vulnerability monitoring
  1. Extend flow probe with plugin
  2. Extend definition IPFIXcol output plugin template
  3. Write down analysis module



Streaming analysis

- Performance results





Streaming analysis

- Running detectors (events/month)
  - Scanning (520117)
  - DoS (324)
  - SSH bruteforce (70)
  - IP spoofing (permanent)
  - Amplification attacks (20)
  - Heartbleed (20 per day)
  - Anomalies PCA (depends)

Streaming analysis

- Advantages
  - Near-real time results & actions
  - Continuous seamless processing
  - Keep data in memory
  - Share outputs
  - Scaling

- Disadvantages
  - Memory management
  - Data copies

FastBit DB

- Developed at the Berkeley Lab
- Columnar storage:
  - Element = Column = File
  - Table = Directory

- Flexible data format
- High performance for large data files
  - Augmented by bitmap indexes

- Allows fast value and range queries
- Strings and binary objects processing

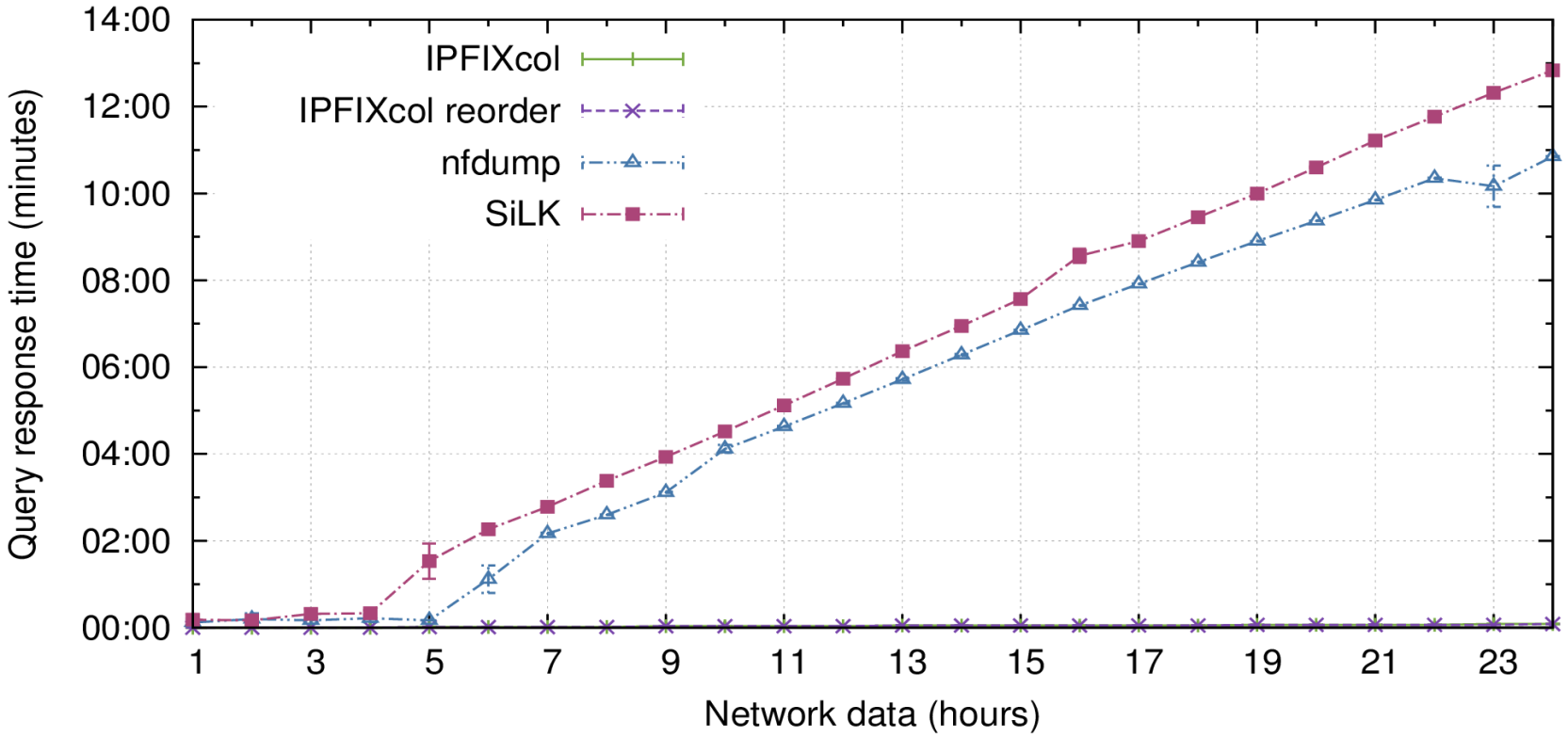
FastBit DB

- Command line data manipulation tools
- C, Java and native C++ API

- **fbitdump**:
  - **nfdump**-like tool
  - Queries over FastBit database
  - Uses C++ API
  - Network oriented (support for IP addresses, protocols, TCP flags, ...)

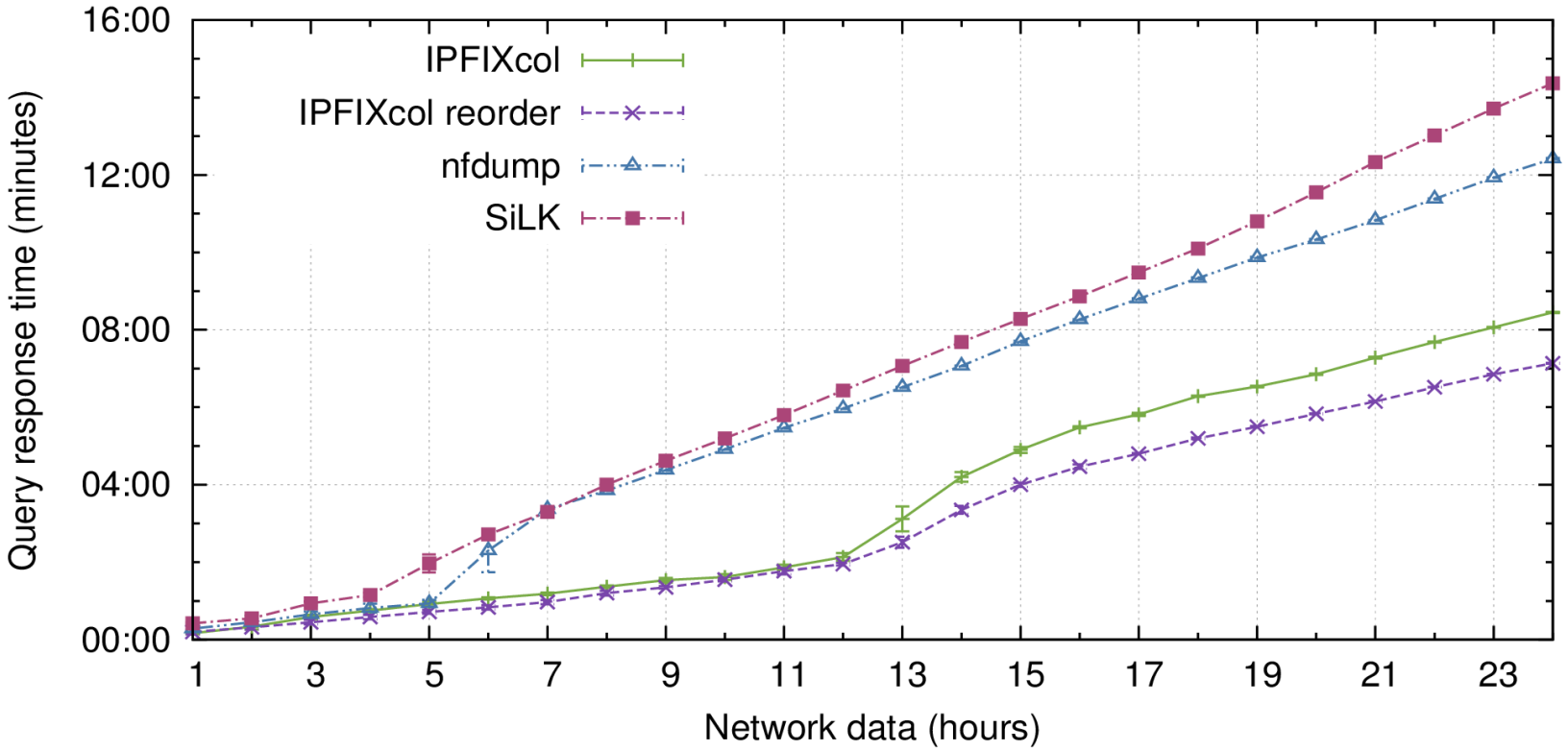
Fbitdump

- Comparison with nfdump and SiLK
- SELECT date start, protocol, src IPv6, dst IPv6, src port, dst port, packets, bytes FROM dataset WHERE dst port = 53 AND ip version = 6



Fbitdump

- SELECT src IPv4, packets, bytes, count(*) FROM dataset WHERE ip version = 4 GROUP BY src IPv4 ORDER BY bytes DESC LIMIT 5



FastBit DB

- Lesson learned
  - Columnar databases are good for flows
  - More *mature* DBs might be considered
- Pros
  - Fast storage access using indexes
  - Easy data manipulation
- Cons
  - Missing efficient aggregation functions
  - Lots of files (table per template)
  - Large indexes

Conclusion

- Alternative flow storage and analysis
  - Columnar DB brings improvement but not solving the problem entirely
- Future work
  - Distributed storage
    - to cope with increasing amount of data
    - to support interactive network forensics
  - Possible way to go is big data processing