Gigascope: How to monitor network traffic 5Gbit/sec at a time.

Chuck Cranor, Ted Johnson,
Oliver Spatscheck

June, 2003
Outline

- Motivation
- Illustrative applications
- Gigascope features
- Gigascope technical details
- Conclusion
Motivation

• Network monitoring is required to maintain network:
  – Security
  – Reliability
  – Performance

• What does that mean?
  – The monitoring tool has to
    • Be flexible enough to allow the operator to react in “Internet time”
    • Achieve high speeds
      – GETH
      – OC48 (2.48Gbit/sec full duplex)
      – OC192 (9.92Gbit/sec full duplex)
    • Be inexpensive
    • Have a small form factor to allow retrofitting
    • Be reliable
    • Be remotely manageable
  – All 7 layers need to be monitored
Existing Solutions

• Use tcpdump and analyze data off line
  – Solution does not scale to even moderate line rates
  – Very expensive due to disk cost and space

• Standard tools such as: SNMP, RMON, Netflow
  – Do not cover layer 7 in a scalable fashion
  – Do not allow changes to the aggregation type at ‘Internet time’

• Proprietary vendor products (sniffer.com, NARUS, handcrafted libpcap tool, …)
  – Do not allow changes to the aggregation type at ‘Internet time’

=> We address those shortcomings with Gigascope, a fast and flexible network monitoring platform
Outline

• Motivation
• **Illustrative applications**
• Gigascope features
• Gigascope technical details
• Conclusion
Case 1: End-to-end TCP Performance Monitoring

• Business Challenge:
  – Customer wanted to monitor the TCP performance of all active clients to identify performance problems

• Current Approach:
  – Netflow and SNMP data do not address end-to-end performance

• Issues:
  – To perform this task accurately, the TCP session state of all current TCP connections has to be tracked to identify active TCP connections
Case 1: Solution

- Gigascope module was used to track all active TCP connections and calculate the average connection throughput.
- Gigascope TCP performance analysis tool will report:
  - Latency
  - Loss rate
- Applicable to both real-time performance monitoring and network/application troubleshooting.
Case 2: Hidden Traffic Detection

• **Business Challenge:**
  – Customer suspected a large volume of peer-to-peer traffic within its network and wanted to more accurately determine the volume of this traffic

• **Current Approach:**
  – Typically Netflow is used to determine P2P traffic volumes using the TCP port number found in the netflow data

• **Issues:**
  – P2P traffic might not use the known P2P port numbers
Case 2: Solution

- Gigascope was used to search for P2P related key words within the content of each TCP datagram
- Gigascope identified 3 times more traffic as P2P than Netflow measurements
Case 3: Custom Monitoring

• Business Challenge:
  – Customer wanted to monitor performance of a critical custom application, in particular:
    • Client application layer metrics (latency, loss)
    • Clients starting and ending the application

• Current approach:
  – Netflow and SNMP data do not address end-to-end performance

• Issues:
  – Commercially available tools do not support analysis of this application protocol
Case 3: Solution

- A Gigascope custom query was written to track the requested metrics in real time and display them in our data warehouse
  - SQL-based query language allowed development in less than 1 week
- In addition:
  - The resulting data was analyzed to reverse engineer the protocol’s performance requirements
  - The data was correlated with Netflow and routing data to perform root cause analysis on service interruptions
Outline

• Motivation
• Illustrative applications
• Gigascope features
• Gigascope technical details
• Conclusion
Gigascope Features and Functions

• Gigascope is a fast, flexible packet monitoring platform
  – Extremely flexible SQL-based interface
• Available for:
  – High performance at speeds up to OC48 (2 * 2.48 Gbit/sec) with a single probe
  – Lower cost at lower speeds
  – Apple PowerBook version available for portable use
• Small size (2U form factor for OC48 version)
• Current libraries include:
  – Netflow compatible records
  – Traffic matrix by site or AS
  – Detection of hidden P2P traffic
  – End-to-end TCP performance monitoring
  – Detailed custom performance statistics
  – More under development…
Gigascope Typical Data Collection Method

Data is gathered using the following steps:

1. Raw data feed is provided by a:
   - Optical splitter
   - Electrical splitter
   - Monitoring/SPAN port

2. Gigascope extracts aggregated records from the data feed in real time

3. Gigascope stores data on local RAID

4. Data is copied in real time or during off peak hours to data warehouse for further analysis.
   - SSH secured back channel
   - Tools allow rate limiting to prevent the Gigascope from flooding the network

5. Data is analyzed and/or joined with other data feeds using Daytona or other tools in the data warehouse

6. Result is displayed, or used for alarming, or to generate customized reports

Optional features:

- Alarms can be generated on the Gigascope directly
- Data can be stored on the Gigascope and only uploaded to data warehouse if needed
  - E.g., additional data can be uploaded after a Denial of Service attack to enable forensic analysis.
Outline

• Motivation
• Illustrative applications
• Gigascope features
• **Gigascope technical details**
• Conclusion
Main Components of Gigascope

- Query compiler: translate GSQL queries into multiple FTA modules (C or C++)
- FTA module: filters, transforms, and aggregates network data
- Application:
  - allocates and configures a set of FTAs
  - collects output of FTAs and “does something” with it
  - Most common application is the “printapp” which writes the query output to disk
- Clearinghouse: manages and tracks all FTAs, manages all IPC, tracks available network input devices
- Device and firmware interface: manages hardware and/or libpcap
Run Time Architecture

- Application processes: perl application, c/c++ application
- HFTA processes: HFTA, HFTA
- Net interface processes: LFTA code dag driver, LFTA ctrl tigon driver, LFTA code pcap driver
- Hardware: dag oc48/gigE, tigon gigE, LFTA code, generic netif
Query Language

- Gigascope queries are written in GSQL
- Similar to SQL
- Support for stream database queries
  - Stream fields can have ordering properties
  - Deduce when aggregates are closed and thus can be flushed to the output stream
- Each query:
  - Receives one or more tuple streams as input
  - Generates one tuple stream as output
- Currently limited to selection, aggregation, views, joins and merges
- Query compiler maps logical query topology to an optimized FTA processing topology
• **GSQL queries receive raw data from low level schemas which are:**
  – Defined on the packet level
  – Use inheritance from lower network layer

• **Current schemas cover**
  – Layer 2: ETH/HDLC
  – Layer 3: IP/IPv4
  – Layer 4: UDP/TCP/ICMP
  – Layer 5-7: Packet data

• **New schemas can be defined by:**
  – Specifying the schema in the GSQL configuration file
  – Providing C parser functions for the fields in the schema

• **RTS and GSQL compiler highly optimize the field extraction**

```c
PROTOCOL IP (Layer2) {
    uint ipversion get_ip_ipversion (snap_len 23);
}

PROTOCOL IPV4 (IP){
    uint hdr_length get_ipv4_hdr_length (snap_len 34);
    uint service_type get_ipv4_tos (snap_len 34);
    uint total_length get_ipv4_total_length (snap_len 34);
    uint id get_ipv4_id (snap_len 34);
    bool do_not_fragment get_ipv4_do_not_fragment (snap_len 34);
    bool more_fragments get_ipv4_more_fragments (snap_len 34);
    uint offset get_ipv4_offset (snap_len 34);
    uint ttl get_ipv4_ttl (snap_len 34);
    uint protocol get_ipv4_protocol (snap_len 34);
    ...
    ...continues...
```
Filtering and Aggregation

Select `tb`, `srcIP`, `sum(len)`
From `IPv4`
Where `protocol=6`
Group by `time/60` as `tb`, `srcIP`
Having `count(*)>5`

- Basic GSQL queries supports filtering and aggregation
- GSQL is similar to SQL and allows:
  - Selection and filtering
  - Aggregation over schema fields which are increasing (e.g. time)
  - Filtering over aggregates
- GSQL queries can be defined over:
  - Low level schemas
  - The output of other GSQL queries or Views
Combining Tuple Streams

Merge P1.time, P2.time
From PacketSum1 P1,
PacketSum2 P2

• Problem
  – Data is collected on multiple network interface cards; for example: to monitor one OC48 full duplex line two OC48 interfaces are required.
  – Some queries are most easily formulated on a single data stream.

• Solution
  – GSQL supports stream merges

• Stream merges
  – Combine two streams with identical fields
  – Maintain the temporal properties of the ‘merge field’ (e.g. time is still increasing in the joined stream)
select S.timestamp, S.sourceIP, S.destIP, S.source_port, S.dest_port, (A.timestamp - S.timestamp) as rtt
INNER_JOIN from tcp_syn S, tcp_syn_ack A

• Problem
  – How to selectively combine diverse tuple streams
• Solution
  – Joins provide an efficient way of selectively combining multiple diverse tuple streams
• GSQL joins
  – Preserve the temporal properties of temporal fields which are used in equalities
  – Current restriction: one temporal field of each input stream has to be used in an equality relationship with a temporal field of the other stream
Integrating Complex Algorithms

OPERATOR_VIEW tcp_perf {
  OPERATOR(file 'tcp_perf');
  FIELDS {
    UINT time (increasing);
    UINT srcIP;
    UINT destIP;
    UINT srcPort;
    UINT destPort
    UINT loss_rate_estimate;
  }
  SUBQUERIES {
    tcpq (UINT time (increasing)
    UINT srcIP,
    UINT destIP,
    UINT srcPort,
    UINT destPort,
    UINT seqNum,
    UINT ackNum
    );
  }
  SELECTION_PUSHDOWN {
    time -> tcpq.time;
    srcIP -> tcpq.srcIP;
    destIP -> tcpq.destIP;
    srcPort -> tcpq.srcPort;
    destPort -> tcpq.destPort;
  }
}

• Problem
  – How to reuse prior and ongoing work in efficient networking algorithms

• Solution
  – GSQL views

• GSQL views
  – Allow user to handcode FTAs
  – Provide syntax and semantic of the input/output datastreams of handcoded FTA to GSQL
  – Optimize automatically ‘surrounding’ FTAs

• Example uses:
  – IP/TCP reassembly
  – TCP performance
Definitions:
uint FUN getlpmid(uint, string HANDLE);

PRED str_regex_match[string, string HANDLE];

string FUN [PARTIAL]
    str_extract_regex(string, string HANDLE);

Example use:
select
    getlpmid(srcIP,’bgp.dat’),
    str_extract_regex(TCP_DATA,’HOST[^\n]*’)
from TCP
where
    str_regex_match[TCP_DATA, ’HTTP1’]

• Each view requires its own FTA and therefore its own thread.
  □ Views are heavy weight
• External functions are a less expensive alternative.
• External functions:
  • Are integrated into an auto generated FTA
  • Can be used as:
    • Functions
    • Predicates
  • Do not have a thread
  • Can have state
  • Can produce one or none output per call but no more
• Examples:
  • String search
  • Longest Prefix Match
An Example Query Collection
Example Deployment

INTERNET

Access Router

Access Router

GETH

GETH

GETH

Customer Equipment

GIGASCOPE

To management

VPN

GETH SPAN PORT

GETH SPAN PORT

GETH SPAN PORT
Gigascope is configured to track:
- Number and IP address of all user
- Detailed performance statistics of 5% of all user including:
  - Client latency
  - Client loss
  - Server latency
  - Server loss
- All ICMP unreachable messages

In this setup Gigascope monitors at peak times:
- 90000 users
- > 1 million packets per second
- > 1.4 Gbit/sec

Gigascope has been operational since 11/2002
One Gigascope benefit is to provide early data reduction through targeted aggregation.

Data reduction rates are application specific.

Typical application:
- App1: Example Deployment
- App2: Detailed Accounting in 1 minute granularity
- App3: Detailed Accounting, P2P detection and TCP throughput monitoring

Remark: All 3 applications are susceptible to DOS attacks.
Conclusions

• Network monitoring requires the processing of streams of network data
  – At high rates (OC48, OC192)
  – Flexibly
  – Inexpensively

• Gigascope is a stream database designed to address this problem