Microsoft

Building a Scalable Telemetry Collection Pipeline

Chao-Chih Chen, Kamil Cudnik, Petr Lapukhov, Edet Nkposong, Lihua Yuan, Yinfang Zhuang

Microsoft

Presentation Outline

Problem Statement

Telemetry data collection
Routing state monitoring
Big Data challenges

Our Solution

Solution components
Collection infrastructure
Storage infrastructure
Data analysis pipeline

Problem Statement

Telemetry data collection

Flow data use cases

Flow Data is: NetFlow, jFlow, sFlow, IPFIX etc

Capacity Planning

Application traffic structure

Forensic information

Billing customers

Collection scale

Thousands of devices (core + data-center)

Up to hundreds of megabits of telemetry data [Gbps in future]

E.g. sFlow data ~ to packet per second rate

Multiple recipients of data

Real-time replication

Different analysis engines

Experimentation in real-time

Routing state monitoring

BGP is the main routing protocol

The only protocol used for **data-center** routing

Core network overlay routing

Keeping it all BGP for simplicity

Need to capture live routing state

Thousands of routers

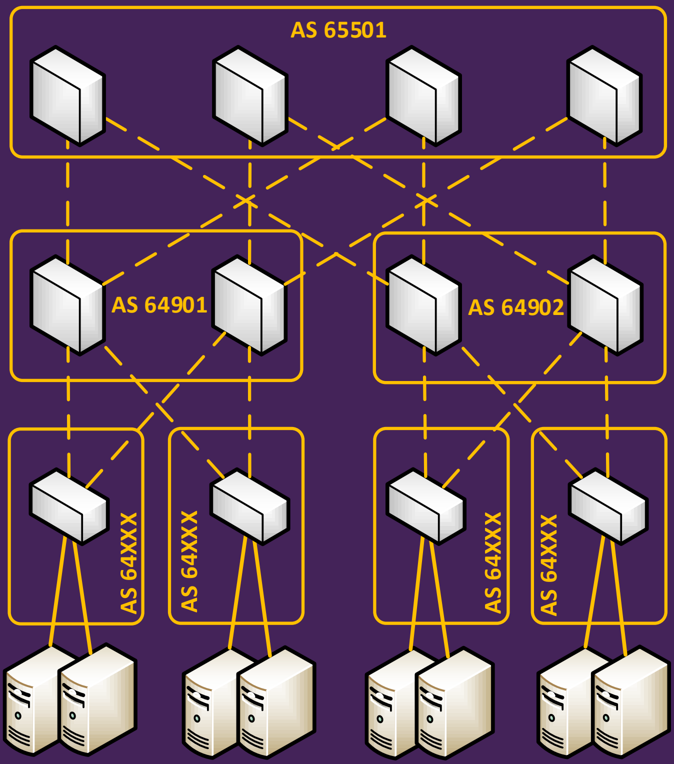
100’s thousands of prefixes

Best-paths and next-hops

Methods for route state monitoring

iBGP peering

SNMP for meta-data



Big Data analysis challenge

Large data-sets

Lots of data being archived

On the scale of **hundreds TB**

Data is commonly append-only

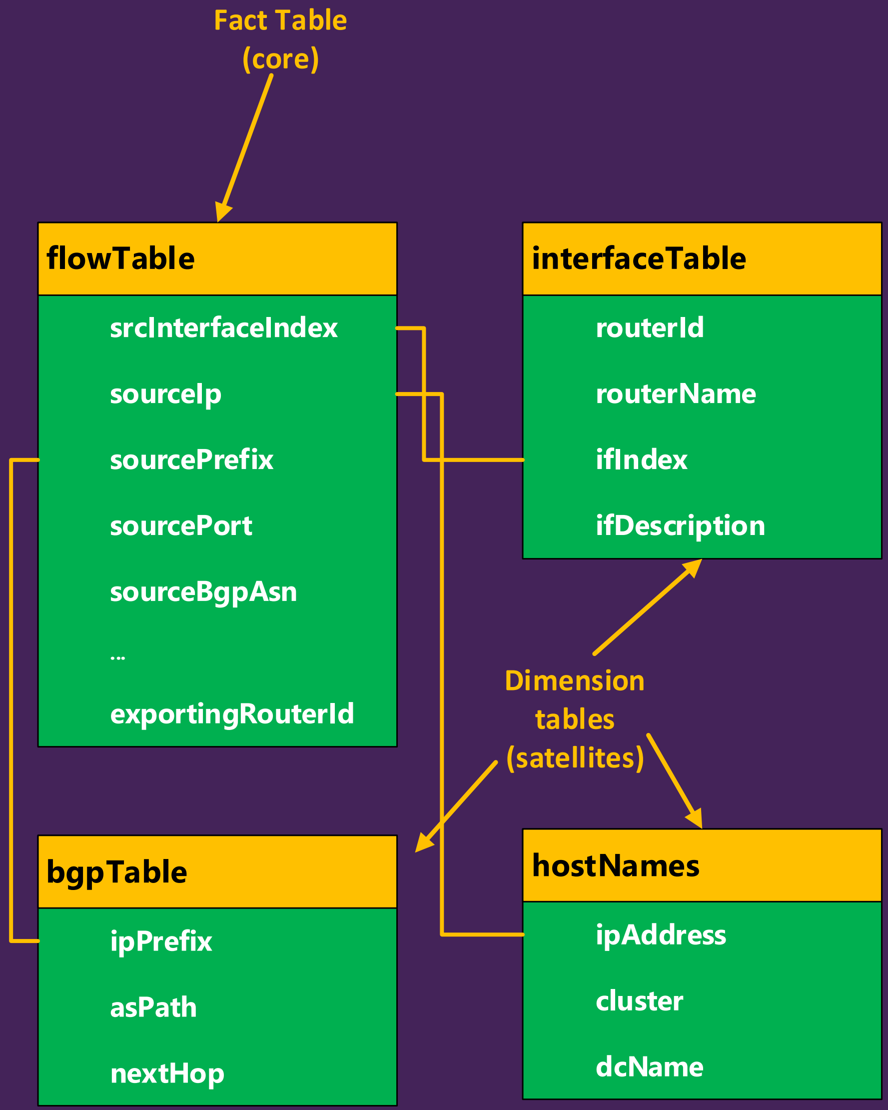
Technical challenge

Need to join large data-sets together

E.g. Netflow on BGP

E.g. joining fact tables on dimensions

Think 'star schema at web-scale'



Collection Infrastructure

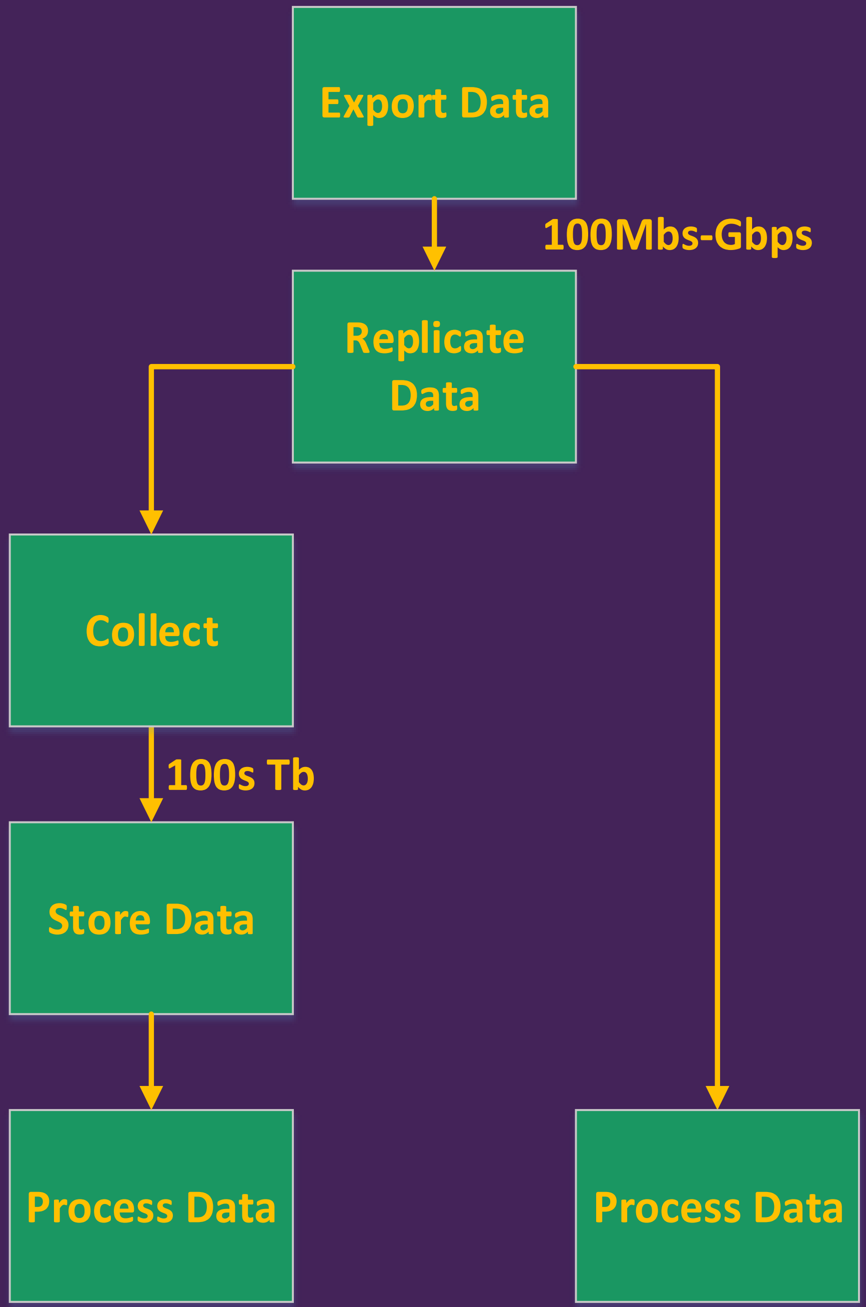
Solution outline

Our goals

Scale to gigabits of telemetry data feed
Minimize cost of infrastructure
Build extensible system
Combine telemetry with other data

Non-goals

Interactive front-end to large data
Full real-time processing system
One size fits all solution



Exporting flow data

Netflow

Peering edge routers

Data center routers

Monitored traffic: 100's Gbps

sFlow

East-West traffic in Data Center

Spine/Leaf/ToR devices

Terabits of traffic

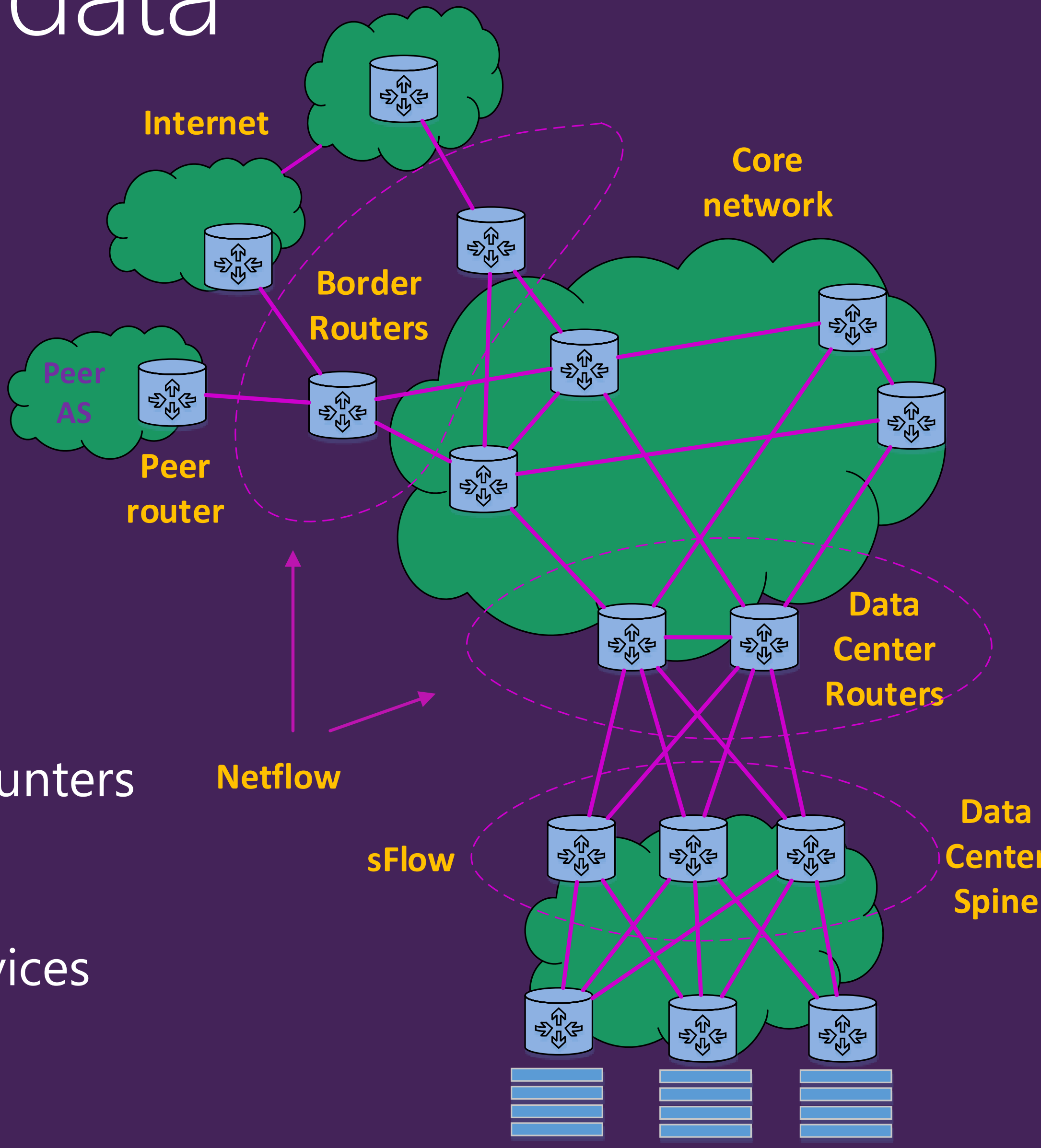
Packet samples and interface counters

Traffic sampling

To reduce load on exporting devices

sFlow is naturally sampled

Random sampled NetFlow



Collecting Flow Data: SLB

Scalable load balancing (SLB)

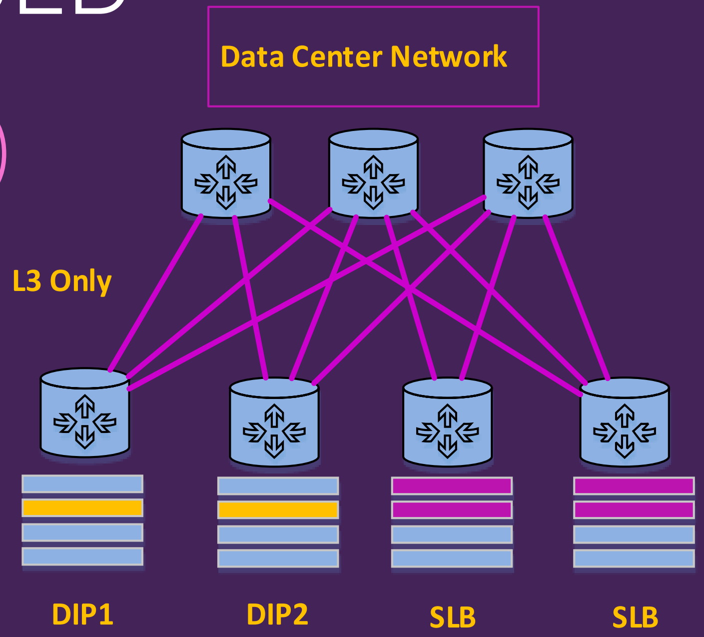
Integrated in data-center network
Scales horizontally by adding nodes
Simple and reliable

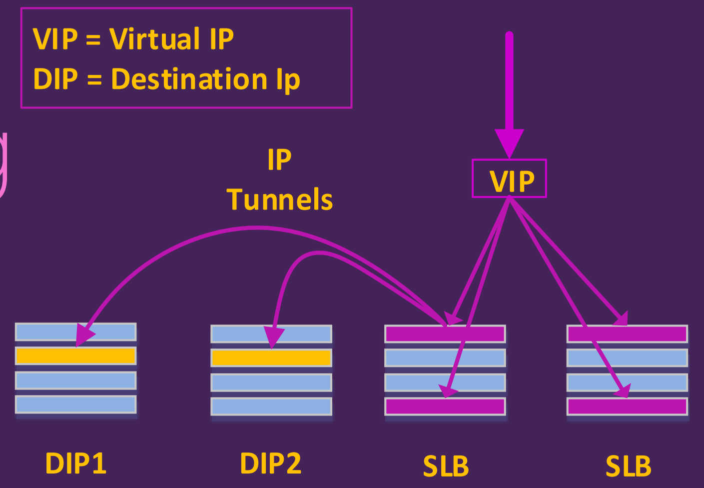
First-stage load balancing

Load-balance the load-balancers
Implemented via **Anycast**
Spray VIP traffic over SLB nodes (2nd stage)

Second stage load-balancing

Stateless load-balancer nodes
Consistent hashing to DIP's
IP tunneling for L3 **Direct Server Return**





Collecting Flow Data: Replication

Anycast VIP

All devices export to global Anycast VIP

Provides geo-redundancy

Simple global load-balancing

Replicating Data

Packets routed to SLB

SLB performs data replication

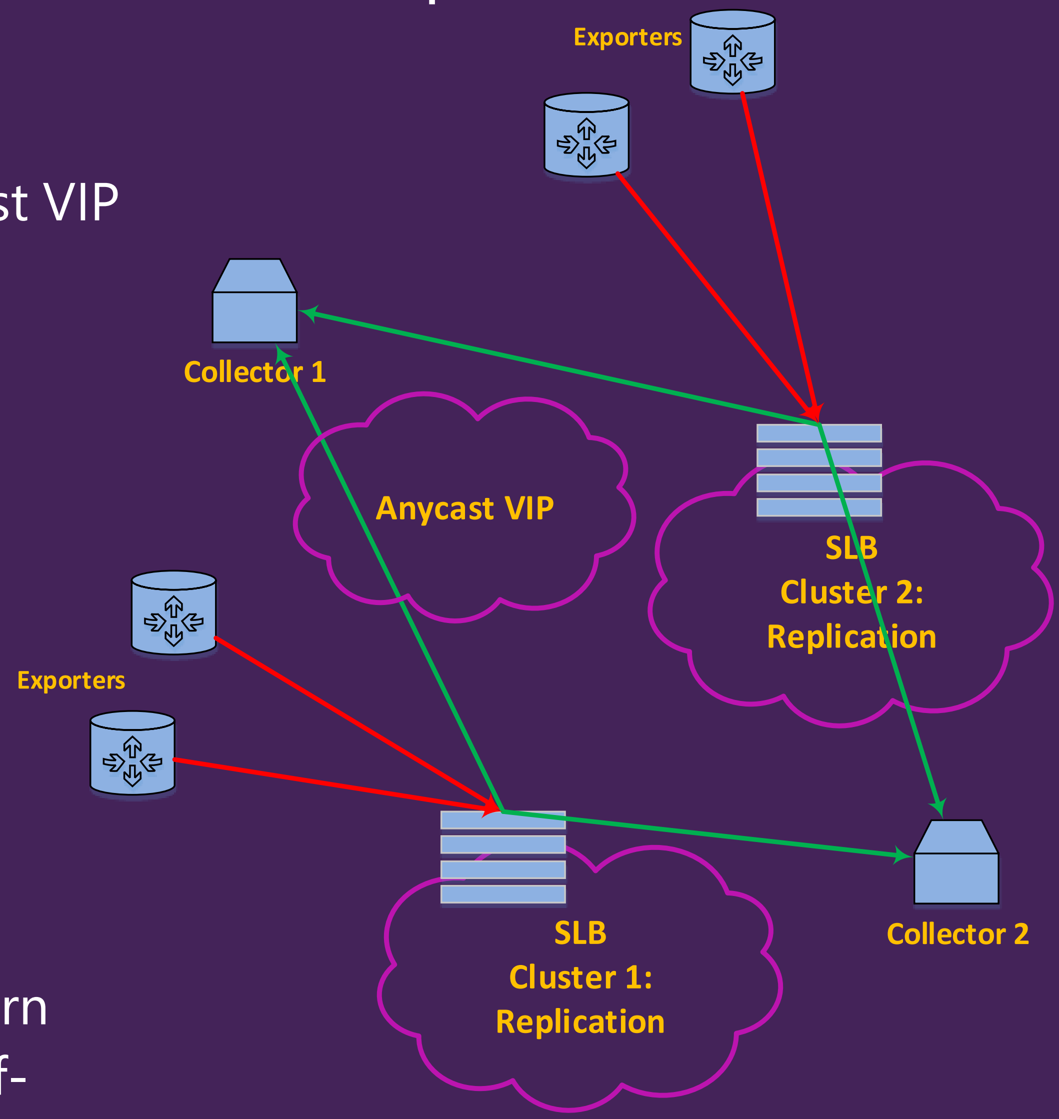
Destination IP re-written for every collector

Distributing data

SLB sends a copy to all collectors

Each collector is behind a VIP in turn

Additional collectors added on self-service basis



Collecting Flow Data: Collectors

Horizontally scalable solution

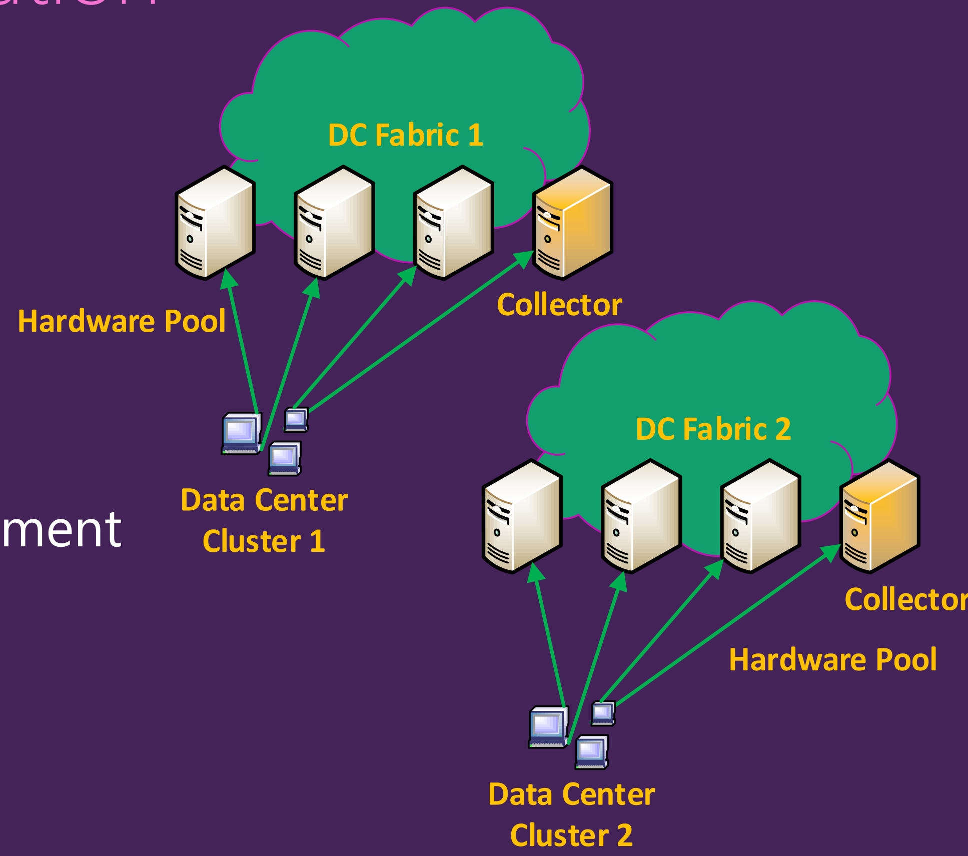
Located behind SLB VIP
Easy to grow/shrink
Servers run sFlow/Netflow software

Fungible infrastructure

Managed by AutoPilot software
Full server lifecycle management
Automated software and data deployment
Web-scale fungible resource pool

Data is automatically partitioned

By means of SLB hashing
Collectors do not need to exchange data
"Shared nothing" architecture



BGP Monitoring

Collect routing state

Located behind SLB VIP

Listen for incoming BGP connections

Always establish iBGP session

Collects BGP UPDATE messages

Does not capture all BGP paths

Sufficient for majority of needs

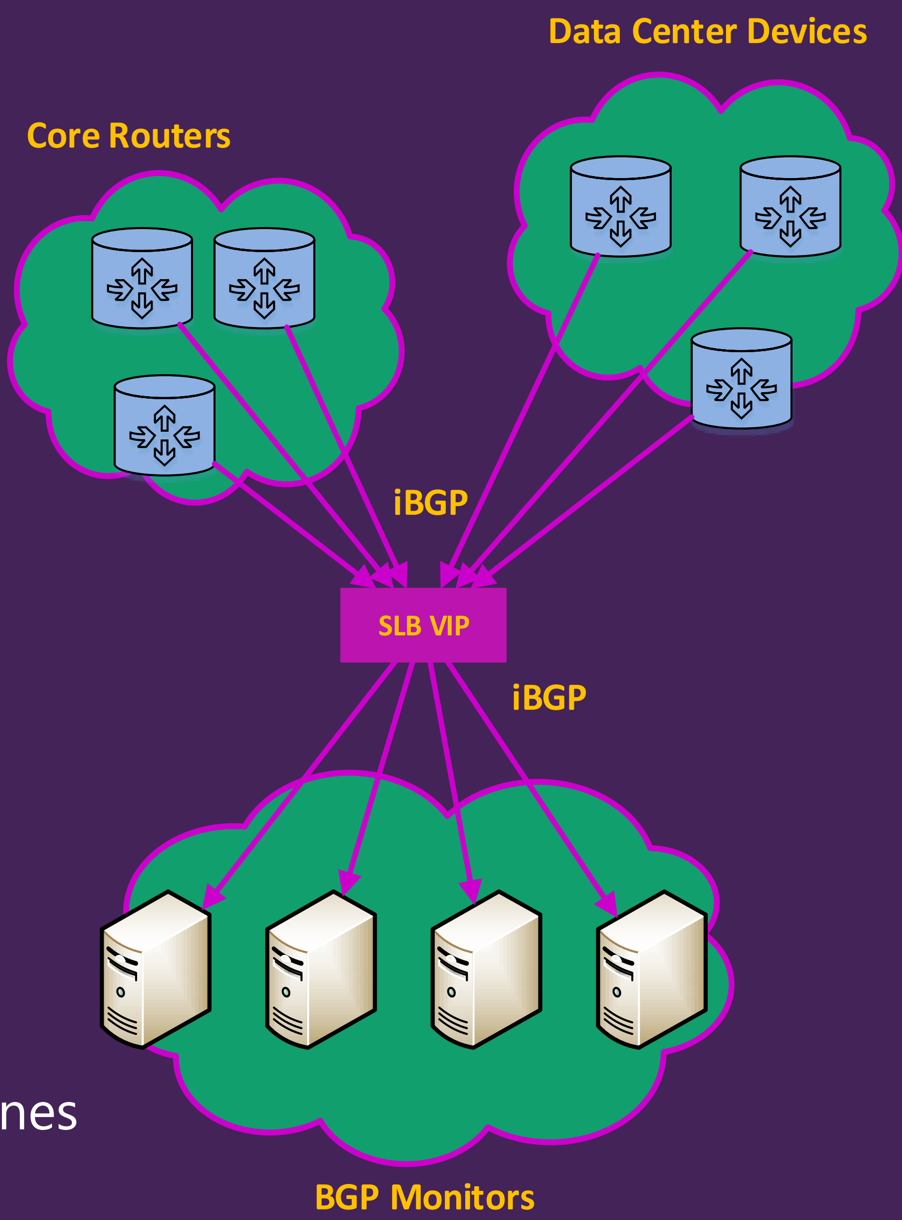
Full BGP table could be built based on UPDATEs

BGP Data Processing

Processed locally by monitoring machines

E.g. to trigger alerts

All data imported in persistent storage



Data Analysis Pipeline

Cosmos: Compute & Storage

Big Data solution for Microsoft Online Services

Based on Map-Reduce idea [similar to Hadoop]

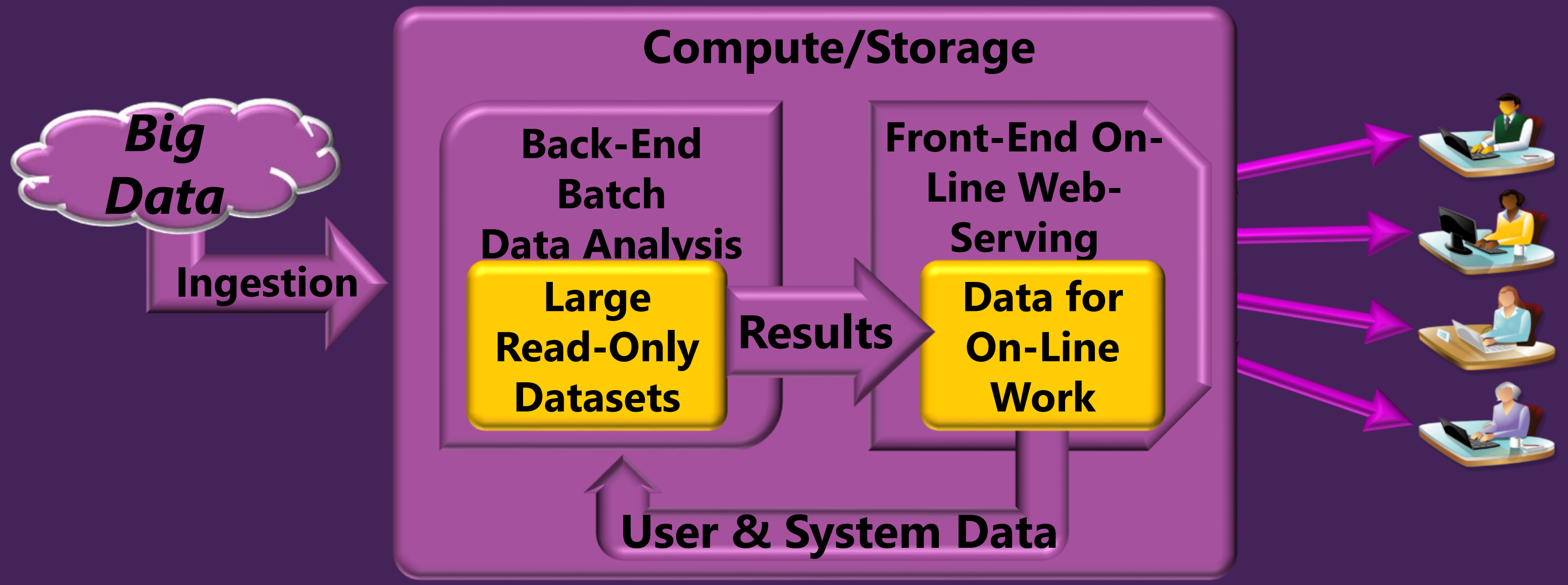
Uses “Dryad” framework (see references)

Provides reliable bulk storage

Generally append-only (streams of records)

Streams partitioned by time buckets

Petabytes of data



Cosmos: Data Ingestion Process

Collectors create local logs

Regular text files, CSV formatted

Applies both to BGP and xFlow data

Collectors import data in Cosmos Storage

Scheduled periodically by a local process

Cosmos Front-End redirects to particular server

All interaction are HTTP based

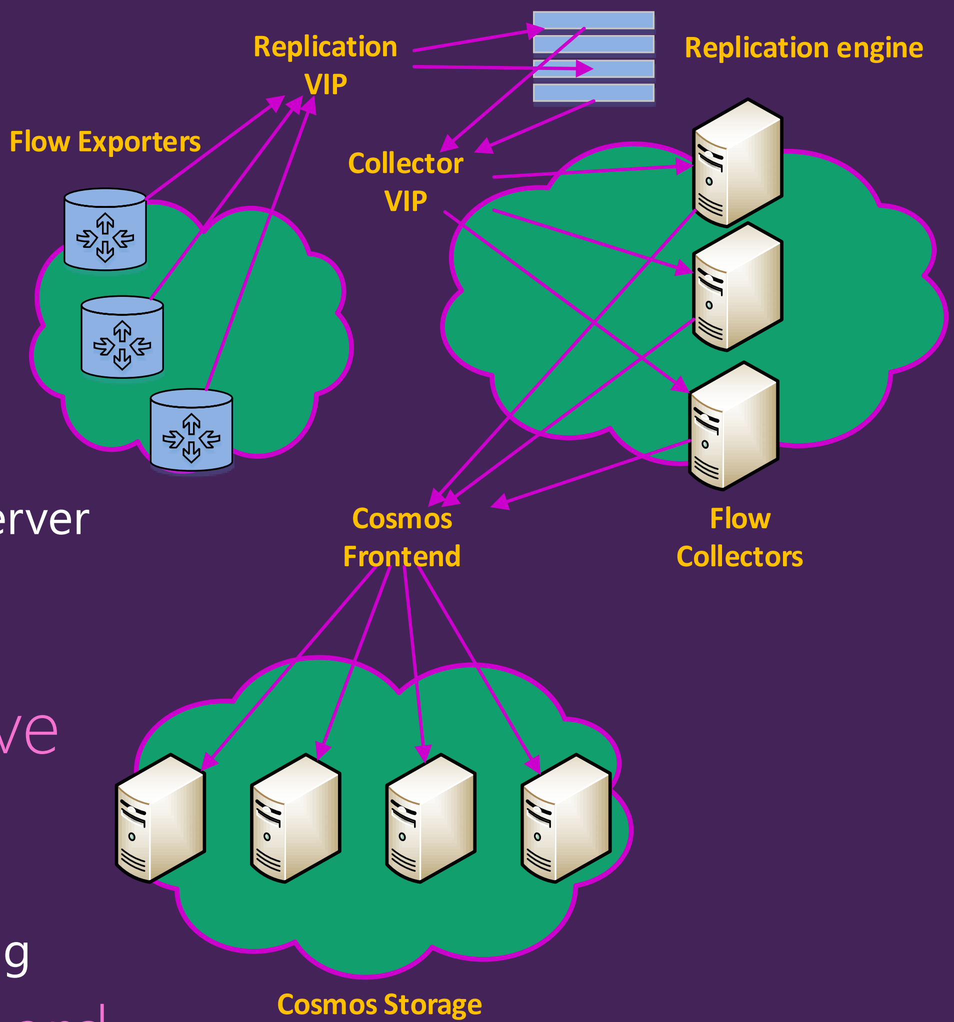
No bottleneck at the front-end server

Data is appended to respective Cosmos “stream”

The only structure is time-bucket structure

Data is stored as “CSV” without any indexing

Monitored via central dashboard



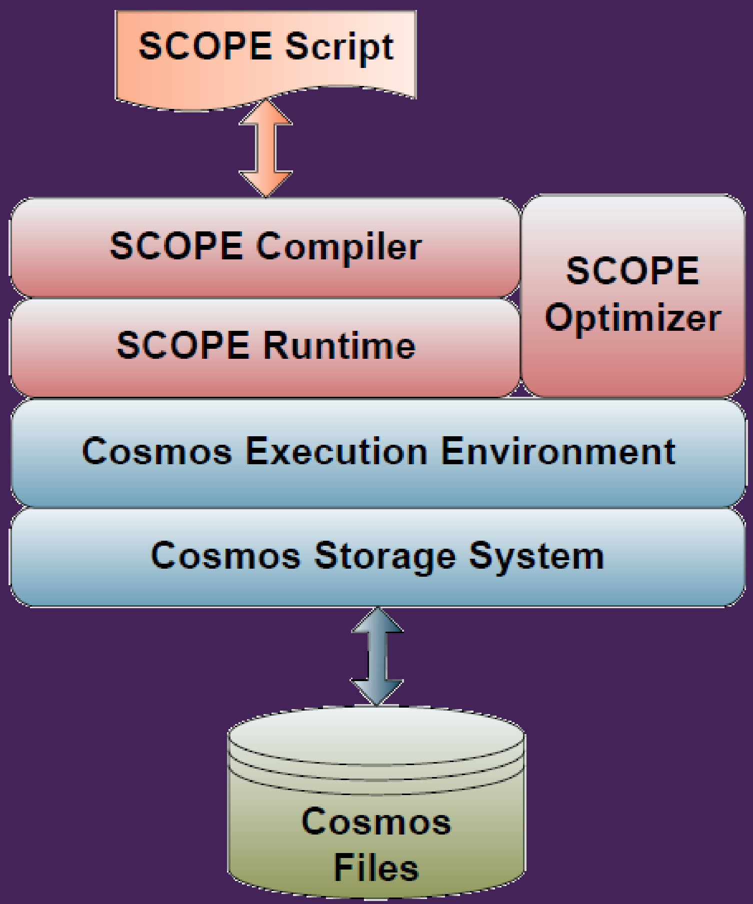
Processing Data: SCOPE Language

SCOPE ::= Structured Computation Optimized for Parallel Execution

- Deliberately similar to SQL in syntax
- Supports C# functions/extensions
- Adds non-SQL operations e.g. REDUCE with custom C# code

```
SELECT query, COUNT(*) AS count
FROM "search.log"
USING LogExtractor
GROUP BY query
HAVING count > 1000
ORDER BY count DESC;

OUTPUT TO "qcount.result";
```



Data Analysis: Input tables

Main Set: xFlow table

timeStamp, 5-tuple, byte/packet counts, ifIndex, duration, exporting router etc

Router interface table

routerId, ifIndex, ifName, ifDescription

Network Topology table

routerId1, routerId2, Latency

sFlow Interface counters

routerId, timestamp, ifIndex, inBytes, inPackets, outBytes, outPackets

BGP Table

timeStamp, routerId, ipPrefix, AS_PATH, NextHop, LocalPref

Grouping factors

Geo-IP/ASN mappings

Internal mappings, e.g. application clusters, data-centers etc

Common Issues and Solutions

Data duplication

Single flow may cross multiple routers

De-duplicated at **time of execution** by AVG()'ing across all devices

In fact this even improves accuracy and gets better sampling rate…

sFlow is flow-less

sFlow only exports single packet information

Flow is "reconstructed" by running SUM() over time-period, e.g 1 min

…Grouped on 5-tuple: all done at job **run-time**

Sampling Accuracy

Resolution is limited by sampling rate: small flow data is inevitably lost

However enough for capacity planning and DDoS attack detection

…Since we only care for intense or long flows

Accuracy validated against interface counters: 5-6% error

Data Analysis: Basic Reports

Top talkers

…Project and select relevant subset of flows

…Group on some factor, e.g. IP prefixes or application environment

…Compute traffic-rate percentiles for each group, select top N

Traffic matrix

Mainly needed for capacity planning

Partition src/dst IP addresses in groups (e.g. Data-Centers)

Compute traffic rate percentiles between groups

Forensic

Who talks to whom – especially if they should not be talking

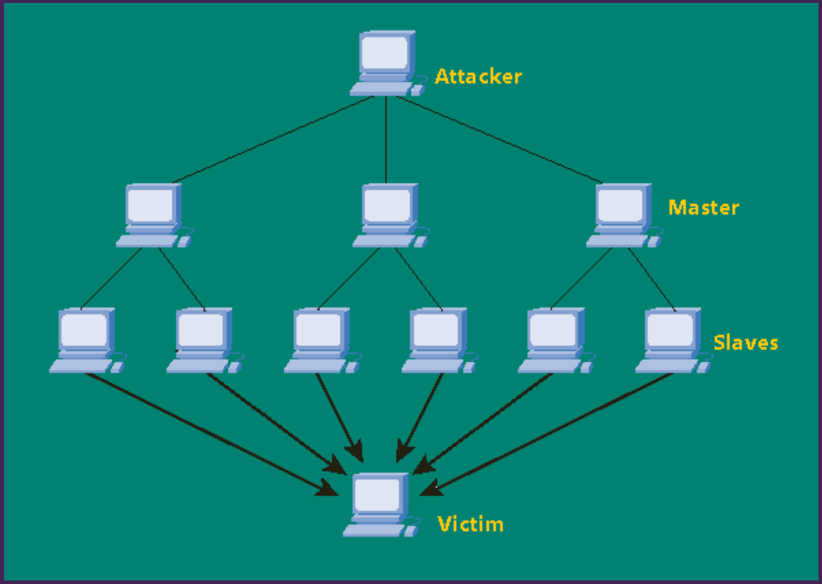
E.g. outbound connections from servers to unknown destiations

Data Analysis: DDoS Detection

DDoS Attacks

Characterized by intense “converging” flows

Commonly TCP SYN/UDP/ICMP packets



Detect Victims

Bin flow data in time intervals

Build traffic rates for each destination:
IP or cluster

Select “suspicious” destinations := “victims”

Correlate with attackers

Correlate victims with flow table by time

Select top sending hosts for each victim

Send information to response system

Data Analysis: Routing Performance

Main Question

How efficiently is routing picking up ingress/egress points?

Metrics are limited, but still useful

Select targets

Set of public IP addresses e.g. web services

Pick up direction: ingress or egress

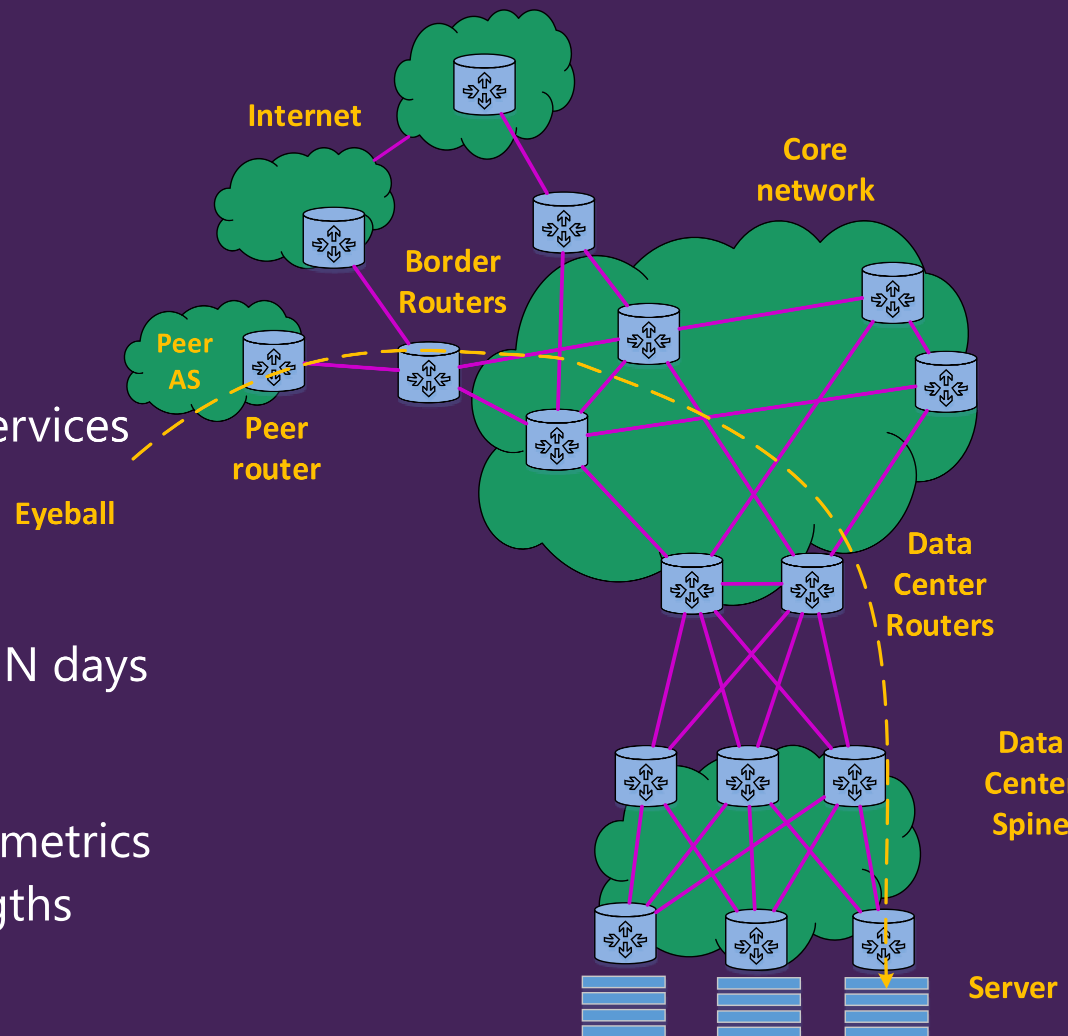
Capture statistics

All flow data to/from the target over N days

Calculate metrics

Distribution of traffic X internal path metrics

Distribution of traffic X AS_PATH lengths



Data Analysis: Edge Routing Optimization

Edge traffic engineering

Multiple paths to same destination…

BGP is not very intelligent in picking those

Find top-talking prefixes on hot links

Move hot prefixes to cold links

Defining where to move

New edge link has to meet some criteria

Has to be cold to pick up new traffic

Has to be close enough to original exit point

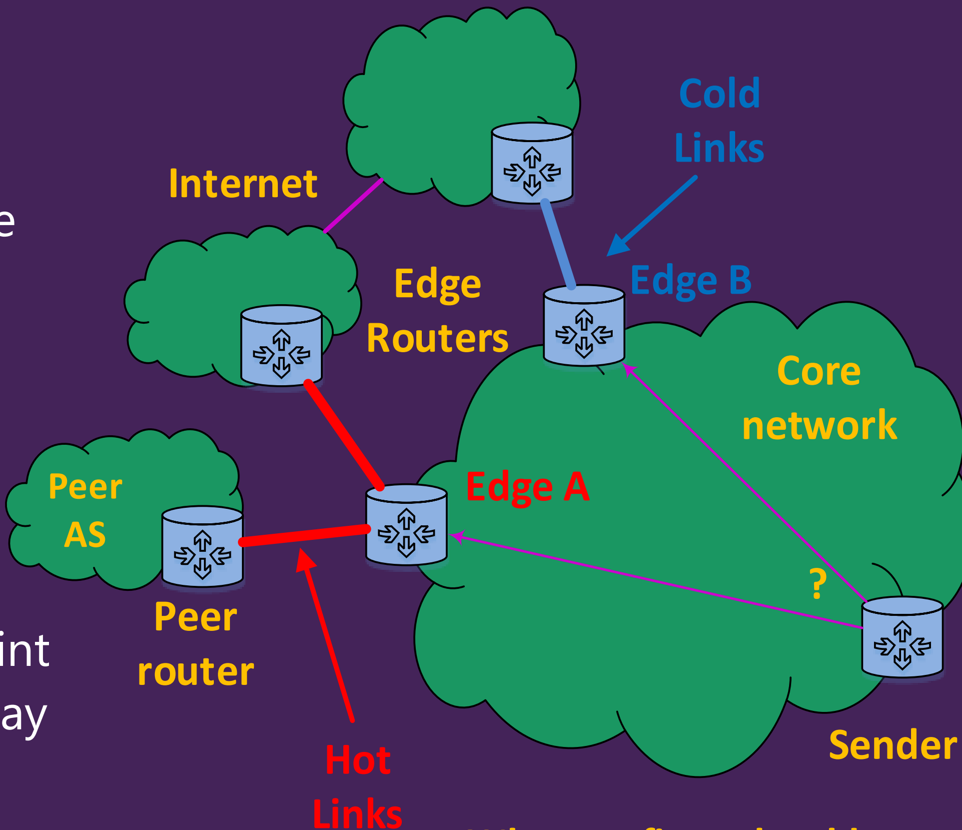
New edge router should not be too far away from destination

Moving the prefix

Either a policy change (LP/MED/IGP cost)

Or injected from central controller (BGP)

One job for all links!



What prefixes should (and can) we move to rebalance traffic?

Summary

Our goal was to collect and analyze very large volumes of telemetry data

Primarily sFlow and Netflow, but includes other exportable data

We integrated telemetry collection system into our infrastructure

Driven primarily by horizontal scalability and cost requirements

Perfect for batch processing, less flexible for interactive queries

Majority reports are still batch in their nature anyways
Real-time reporting applications run on servers
System allows for advanced analytics by integrating other data sources

How does that relate to me?

Big-Data is not just for big companies

You can build your own Hadoop cluster

Could be as small as few servers in one rack

Horizontally-scalable load-balancing is not hard

You can do it using off the shelf equipment

Implement Layer 3 Direct Server Return

Or simply use Anycast and ECMP since consistency is not a big deal

Could be used as a fist step to Big-Data

Once you have one applications, many will follow

E.g. dump all your telemetry to Hadoop

Microsoft

References

Autopilot Reference

http://research.microsoft.com/pubs/64604/osr2007.pdf

COSMOS Reference

http://research.microsoft.com/en-us/events/fs2011/helland_cosmos_big_data_and_big_challenges.pdf

Dryad Reference

http://research.microsoft.com/en-us/projects/dryad/

SLB Reference

http://www.nanog.org/meetings/nanog47/presentations/Monday/Chen_RouteControl_N47_Mon.pdf