Characterizing Large DNS Traces Using Graphs

Balachander Krishnamurthy,
http://www.research.att.com/~bala/papers

(joint with Charles D. Cranor, Emden Gansner, Oliver Spatscheck)

Characterizing DNS traffic

Why DNS?

- DNS traffic has grown rapidly
- First due to growth of Web and the large number of URLs accessed
- Made worse by CDNs abusing DNS via overlay network
- Although not always useful from client latency POV

Motivation

- Characterizing DNS traffic helps identify agents
- Clients, local DNS servers, authoritative DNS servers
- ... and interaction between them
- Identify any anomalous hosts
- Through an entirely automated process

Study

- Extract IP addresses suspected to be involved in DNS
- Categorize into clients, LDNS, ADNS with high probability
- Validate categorization
- Allow incremental addition of new data sets

netflow data

- netflow (Version 5) data being gathered at peering links in AT&T
- Collection of IGRs terminating access links
- Unidirectional packet sequence
- Twenty fields in flow data gathered
- Voluminous; over 200G a day compressed

Fields in netflow data dumped

Source and destination IP Address
IP Address of next hop router
Input and output interface index
Packets and octets sent in duration
SysUptime at start and end of flow (cheap clock on linecards)
Source and destination port number
Cumulative OR of tcp flags
Protocol identifier (6=TCP, 17=UDP)
IP type-of-service
Originating AS of source and destination address
Source and destination address prefix mask bits

When is a netflow record dumped?

- Flow closes (say, TCP FIN is received) or quiescence in endpoints for a period of time (15/30 seconds)
- Continuous activity between endpoints (30 minutes)
- Router cache full

Intra-record exchange obscured by netflow (multiple messages between same source/destination are aggregated)

Among dumped src/dest ip address, port numbers, protocol, number of bytes and packets, OR of TCP flags are fields primarily used by us.

Identifying DNS

Normal DNS traffic

- Client query (often UDP, can be TCP)
- Server response
- Zone transfers - synchronization between primary and secondary DNS servers for consistency (on TCP) (per RFC 1033)

We gather data on port 53 (src or dest, UDP or TCP)

Both src and dest port can be 53!

Graph representation

- Each IP address extracted from netflow is a node
- Edge: request/response or zone transfers between nodes, head of edge associated with port 53 IP address
- IP1 -> IP2 (request from IP1 to IP2 or IP2 to IP1 reply, IP2.port = 53)
- IP1 <> IP2 (src and dest port are 53, older BIND) either IP1 or IP2 could be a DNS server
- Graph may have both such edges for given IP1, IP2
- Weight of edge is packet count in exchange

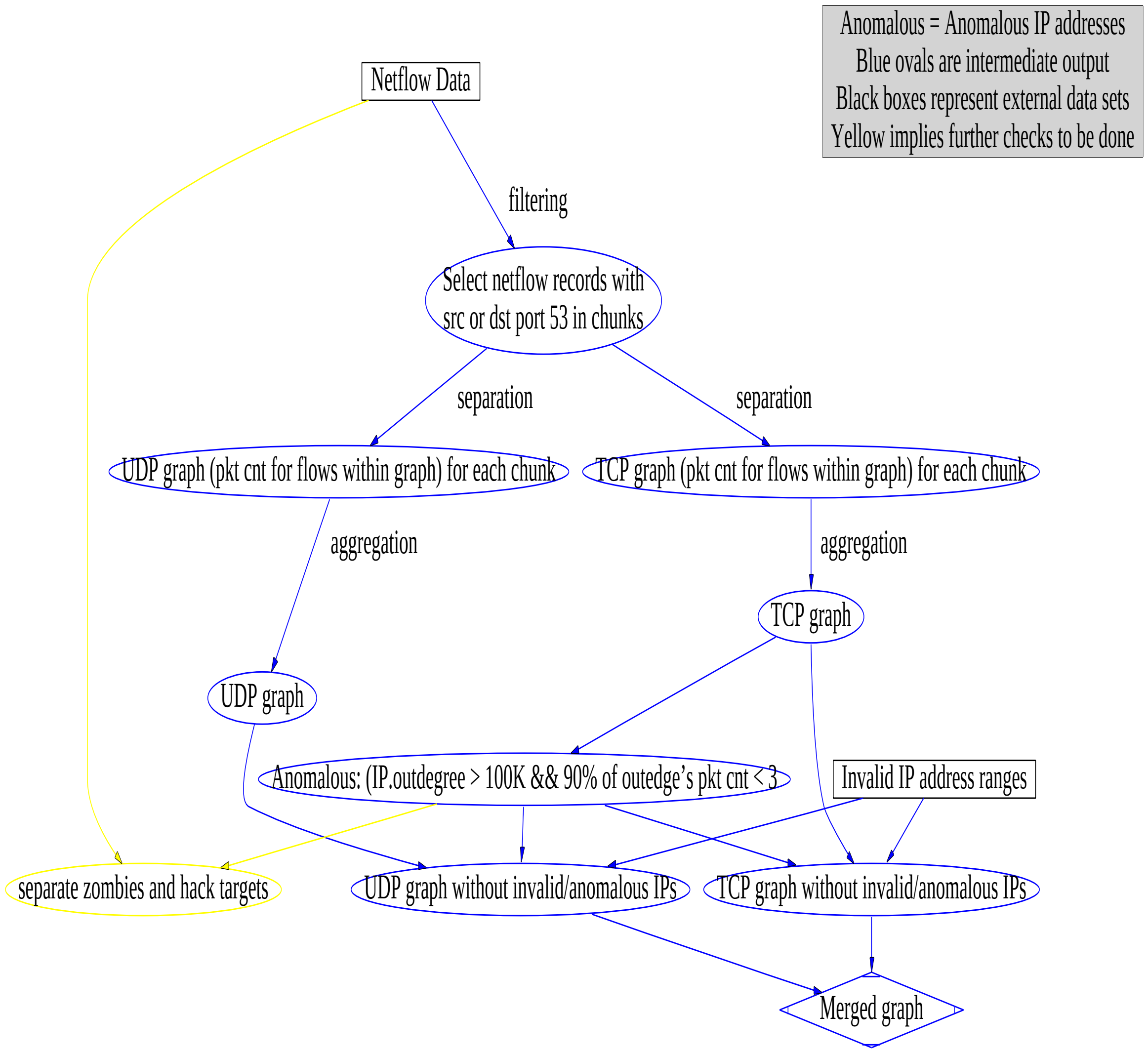
Create separate graphs for UDP and TCP

Graph analysis

- If interesting graph attributes are present, see if they have semantic import. Provides consistency check.
- Node attributes: in-degree, out-degree
- Edge attributes: bi-directionality, common edges in UDP/TCP graphs
- Check for presence/absence of patterns e.g., large in/out-degree

Distribution of connected components consistent with large sparse random graphs (many small components plus 1 monster component containing much of the graph). Smaller components tended to be a collection of nodes pointing to a single node, which in turn points to a set of nodes.

Graph conversion process



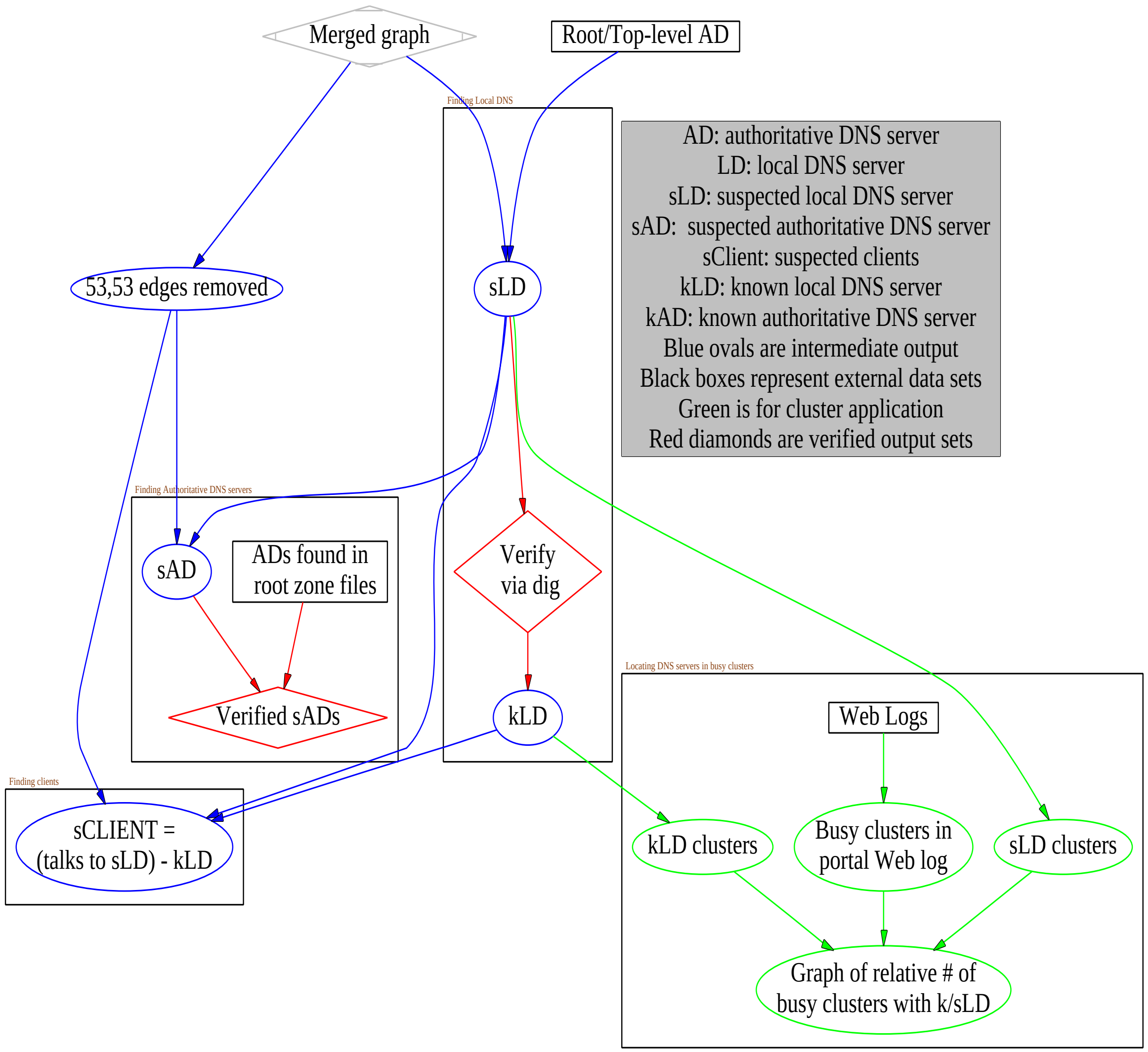
Anomalies

- Short TCP flows: since we may see only one direction of flow, any TCP flow less than 3 is odd
- Multiple short TCP flows with a node in common is very odd
- We found several: zombies, attack targets, misconfigurations, RTT measurement candidates
- Invalid address ranges: 0.*, 10.*, 192.168.*, 172.16.*–172.31.*
- Easy to separate such nodes and related edges from graph

Additional external data

- Set of known root servers (*.ROOT-SERVERS.net), authoritative servers for `.net`, `.com`, `.org`, `.mil`, `.edu`
- NS records in many top level generic and national domains
- BGP snaphots from 15 places
- Web server logs from portal site

Applications of merged graph



Identifying Local DNS servers

- Local DNS server contacts pre-configured set of root servers and refreshes list to obtain IP addresses of generic TLDs
- All nodes that are tail of edges to nodes in the above set are suspected local DNS servers
- Run *dig* to verify sample and generate Known LDNS server set. (ACLs prevent many from replying)

Note that a LDNS may also be configured to be an authoritative DNS server and it is hard to identify them.

Identifying authoritative DNS servers

- Authoritative DNS server is authoritative for a set of domains
- Receive many queries but do not generate (other than to root server at start up)
- Nodes with zero out-degree and head of edges from LDNS nodes are suspected authoritative DNS servers
- But first remove 53-53 edges from merged graph since we do not know for sure which is a LDNS.
- Compare against list of top level/national domains

Identifying clients

- Clients should have modest out-degree
- Gather nodes which are tails of edges whose heads are in suspected LDNS servers and remove known LDNS servers
- Also, as in AD, remove 53-53 edges

Using network-aware clusters

Quick intro to clusters (Krishnamurthy and Wang, SIGCOMM 2000)

- Group IP addresses that are non-overlapping, topologically close, and under common administrative control
- Perform longest prefix matching on addresses against union of collection of BGP prefixes
- Classify addresses with same longest matched prefix into a cluster
- Validate via *nslookup* and *traceroute*

Application of clusters

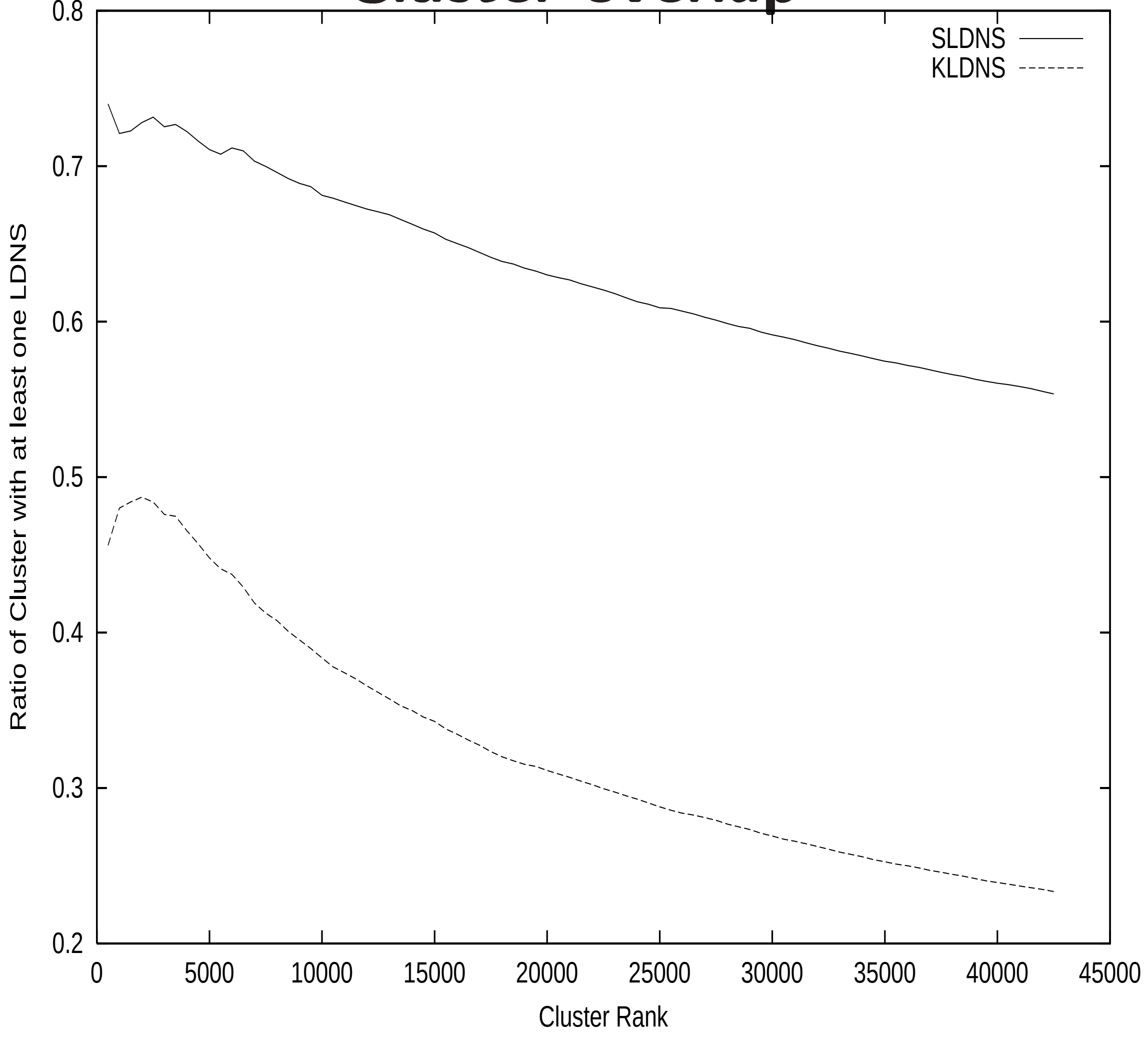
- Cluster IP addresses in suspected local DNS server list and known local DNS server list separately.

- See how many of these clusters overalp with set of busy clusters of the Web clients.

- Plot percentage of matches found in increasing fractions of the set of busy Web client clusters gives a relative measure of busy clusters that include a suspected/known local DNS server.

- This is a lower bound on the number of local DNS servers present in these clusters.

Cluster overlap



Advantages of graph representation

- Can use small graph library to examine out-degree, edge weight, remove bi-directional edges, and for finding node neighbours
- Each few tens of lines of code, runtime O(n*logn)
- Entire library 300 LOC
- Low runtimes

Graph sizes

	Edge count	Kilobytes
Daily UDP data	257,339,100	7,832,896
Daily TCP data	22,435,114	691,057
Merged UDP graph	125,253,867	1,503,046
Merged TCP graph	21,949,654	263,395
Reduced UDP graph	125,204,439	1,502,453
Reduced TCP graph	7,842,576	94,110
Merged TCP+UDP graph	132,849,091	1,594,189
Merged TCP+UDP graph without 53-53 edges	76,224,714	914,694

DNS graph protocol breakdown

	nodes	edges	packets
UDP raw	5,391,685	125,253,867	7,520,127,483
Without illegal IPs	5,374,094	125,204,439	7,506,734,902
% Reduction	0.33	0.04	0.18
TCP raw	19,732,072	21,949,654	149,220,358
Without illegal, anomalous IPs	6,793,678	7,842,576	124,327,321
% Reduction	66	64	17

Characterizing addresses

Suspected LDNS	564,077
SLDNS tested via *dig*	224,348
Known LDNS	58,527
Suspected Authoritative	2,919,455
SAD with outdegree=0	2,717,025
Known AD	217,721
Suspected Client	3,178,981
SClient with Known LDNS removed	3,129,037
SClient with Known LDNS removed and with indeg=0, outdeg < 4	2,298,988

Ongoing and Future work

- Ongoing: Using similar methodology to examine multimedia traffic
- Examining temporal properties (anomalous addresses activity is spiky)
- Examining if random graph's properties have implications in the network context