Danish Farid – 19100172 – NSG Research Intern

The Data available to us in the network trace files is the following:

Packet Sampling Rate: **1:30,000**

USEFUL	***NOT USEFUL***
Timestamp -> *Time at which packet was sent*	Anonymized src/dst hostprefix
Packet length -> *Packet payload*	Anonymized src/dst Pod
Anonymized src/dst IP	
Anonymized src/dst L4 Port	
IP protocol	
Anonymized src/dst Rack	
InterCluster	
InterDatacenter	

To understand the usefulness of the data that we have provided to us in the FbFlow network traces, it is important to establish the ground truth. Is what we have enough to "emulate" what the case is within the actual data center?

To confirm this, the only other known source of information about the Facebook DC is within the paper, the numbers and graphs there. Created after running their data through what would have been, extensive data processing.

There is a total of three graph types in the paper that we can use our data to reasonably emulate and then use to compare and perhaps reach a conclusion about the usefulness of our own data.

1. **Data Flow Rate** in Webservers per second
2. **Packet Size** Cumulative distribution
3. **Flow Size** Cumulative distribution

Other given graphs in the paper include Flow Duration cumulative distribution. This is something I cannot create with our traces without more information in them.

My aim with this short report is, to the best of my ability, recreate the graphs given in the paper with our sampled data and then to compare them. To see if whether missing data causes any OBVIOUS problems with trying to pull generalizations from the graphs.

If for instance, the real graph is an exponential one and ours looks more like a straight line, there is then of course an obvious issue, and constraint. However, if the data sampling isn't too much of a problem then our graphs will mostly be "close enough". We'll be able to tell with a little more confidence soon.

Ahead is the analysis, graph by graph. This is followed by a conclusion of whether we can, within margin of reasonable error, say that our data is "dense" enough to be able to represent the real thing, and that sampling does not remove fundamental intrinsic characteristics from the information.

All my graphs were made from data from the 21st hour of trace information pulled from 1500 of the total of 2700 files available to us. Meaning, the actual graphs may have more information to show still when created with information from all the 2700 files.

I parsed the 1500 files, pulling traces from:

1. The first 120 seconds
2. The first 400 seconds

Saving them to two different files for later access.

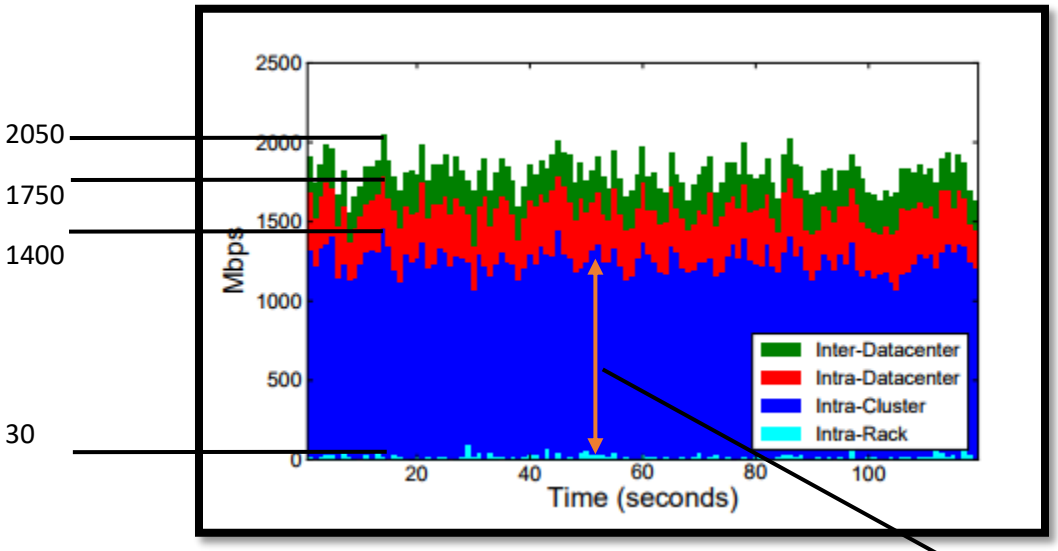
The Python Pandas, Numpy and MatPlotLib were extensively used for all of the following:

==========

1. **PAPER GRAPH 1 ->** Data Flow Rate in Webservers per second

   This graph is of the web server traffic by one of the front-end web clusters

I suspect that the data here is cumulatively represented rather in each type being independent/unique



Graph from the paper defining bitrates

Suspicion:

THIS IS **CUMULATIVELY SUMMING** DATA?

We can tell this cumulative since the InterCluster data is the largest and most prominent data type as per the paper, but only if we look at it as a CDF

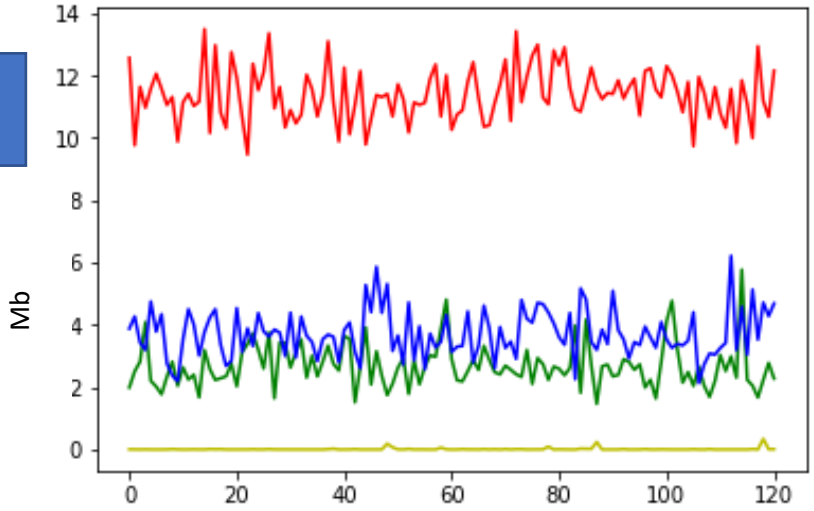
- ➔ My hunch here is that they have allowed for data overlap. Meaning, InterDataCenter traffic contains all traffic, intra rack and intra data center as well (cumulative, basically)
- ➔ If we take out the differences between any two subsequent categories together, we get their actual values which ties in well with my graph on the next page:

Cumulatively the different values add up just fine, meaning we're probably dealing with a **CDF here**

Equivalent Graph recreated from the data in Files

- ➔ Graph 1 and 2 here, were created with data from different times in the 24 hours
- ➔ **GRAPH 1** is just a regular graph, **not a CDF (This was made weeks ago)**

Max Values -> "IntraCluster"



Graph - 1

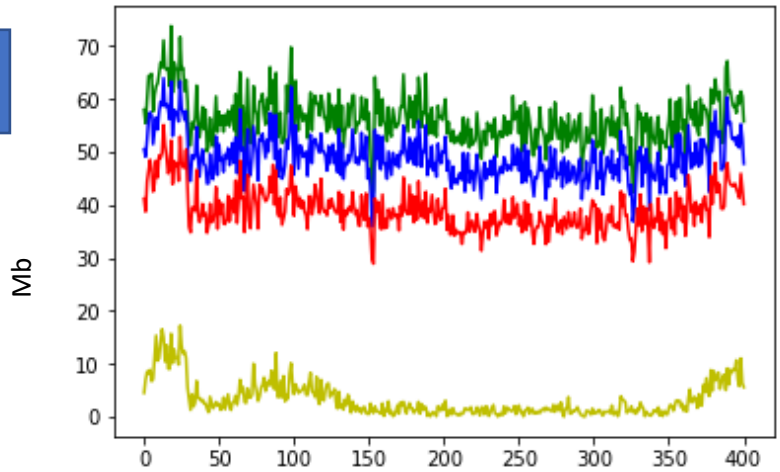
NO CUMLATIVE SUMMING

120 seconds

Data from 500 Files

RED: IntraCluster Traffic
BLUE: IntraDataCenter Traffic
GREEN: InterDataCenter Traffic
YELLOW: IntraRack Traffic

Max Values -> "InterDataCenter"



Graph 2

WITH CUMULATIVE SUMMING

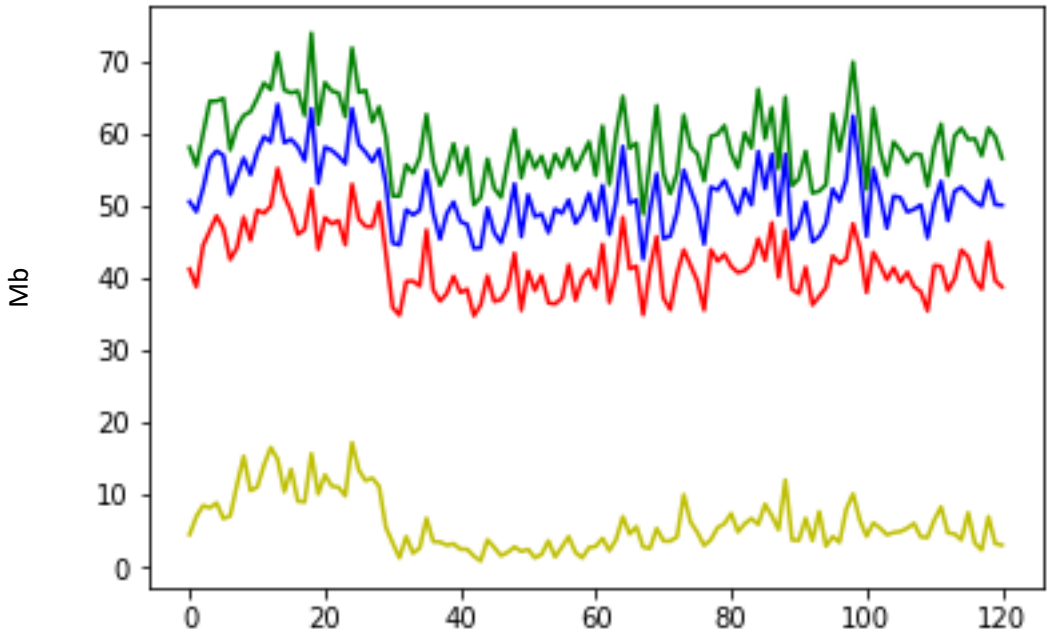
400 seconds

Data from 1501 Files

RED: IntraCluster Traffic
BLUE: IntraDataCenter Traffic
GREEN: InterDataCenter Traffic
YELLOW: IntraRack Traffic

- ➔ **Graph 2** was made when I realized that the graph in the paper didn't show the correct data and was thus probably cumulatively made. Graph 2 confirms it by looking closer to it.
- ➔ While, in **Graph 2** the values on the MegaBits axis are not in the same range or even remotely close to the values in the actual graph, the trend is being reasonably followed. Let's also not forget that the source data itself is technically different for both graphs too.

Another cumulative graph was made exactly like the second one above, except only for 120 seconds to exactly match the one from the paper, this is **Graph 3**



Graph 3

With Cumulative Summing

120 seconds

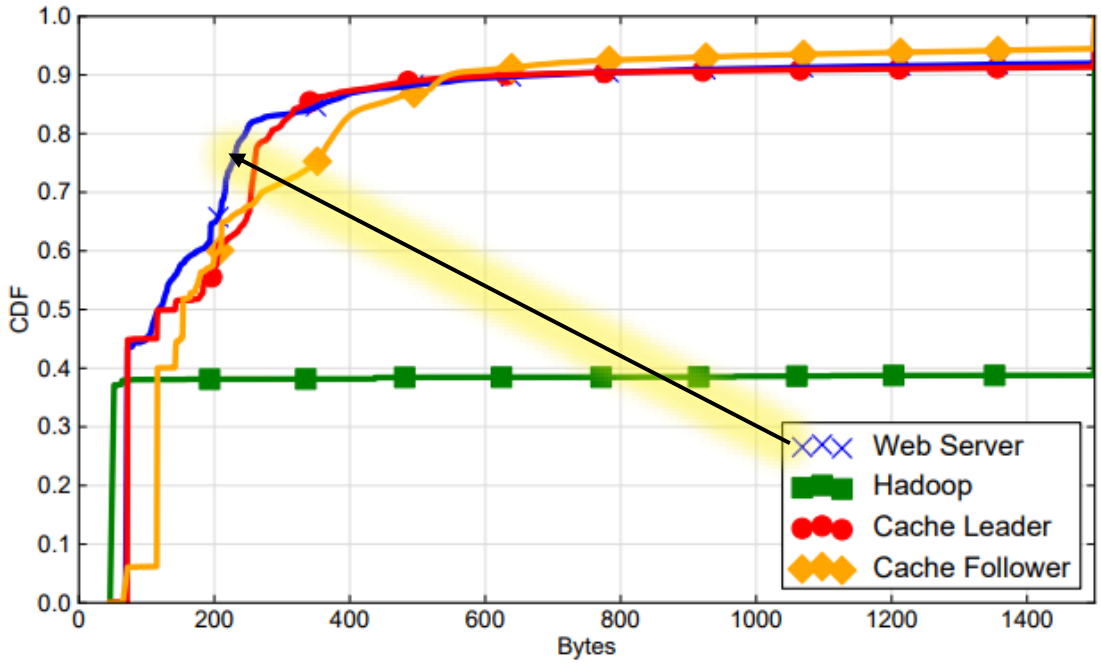
Data from 1501 Files

RED: IntraCluster Traffic
BLUE: IntraDataCenter Traffic
GREEN: InterDataCenter Traffic
YELLOW: IntraRack Traffic

Again, there is good correlation with trends, but of course the values are a lot less.

Paper Graph 2: Packet Size Cumulative Distribution

➔ We're only really interested in the blue line here



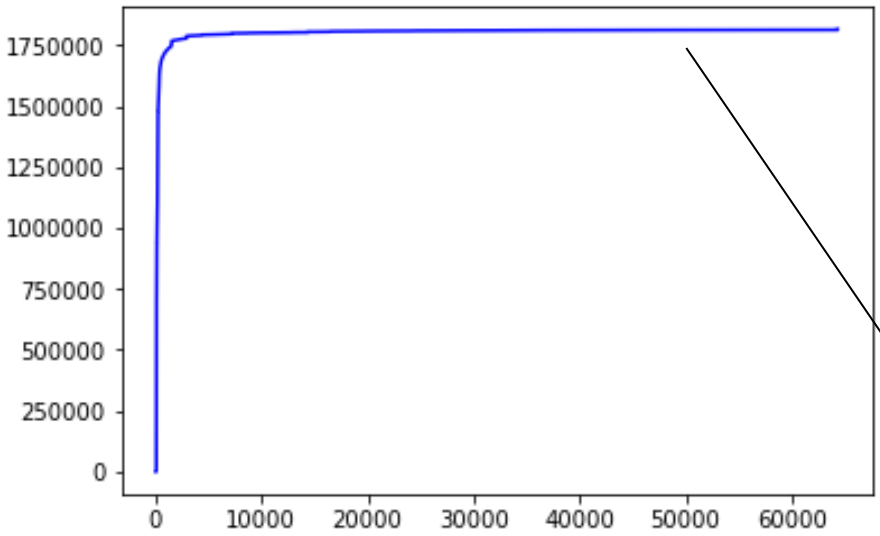
Graph from the paper showing Packet Distribution

Limitations of my graphs created from the data:

- ➔ Since we're doing our processing on local machines, we cannot create a packet-wise distribution for all the packets over the entire 1500 file range for the complete trace duration.
- ➔ Of course, the obvious limitation of only having 1 in every 30,000 traces doesn't help either

To plot this, I took all the packets sent in the first 120 seconds of the 21st hour of trace data:

- ➔ This data contained packets that ranged from 30 bytes, all the way up to 64334 bytes, so the CDF looked like this, the few large packets caused extreme skew.



Graph 4

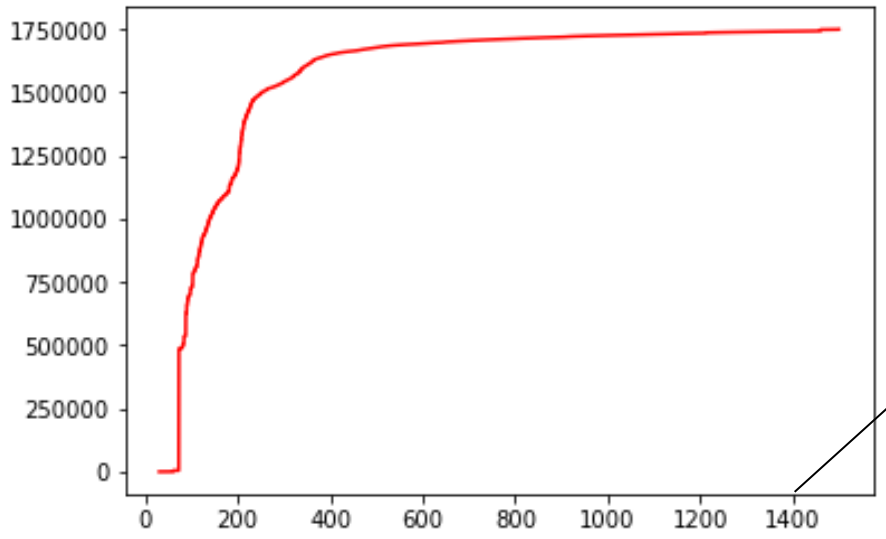
PacketSize Distribution

➔ Data from 1501 Files

A few very large packets are skewing this

120 seconds of trace data from Unix Time -> [1475380800] till [1475380920] contained 1814731 traces

- ➔ Cutting down to 1750000 values however (rest ignored), allowed me to get all the packets within the same range as the graph from the paper – Data Ranged from 30 - 1500 bytes



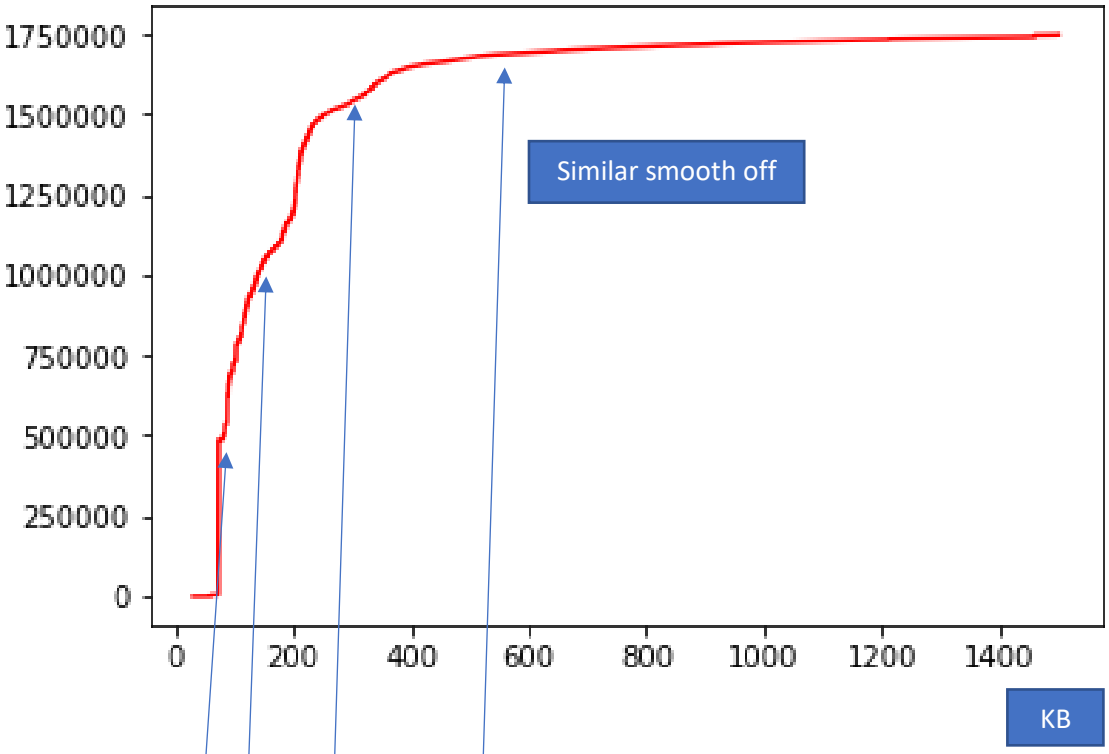
Graph 5

PacketSize Distribution

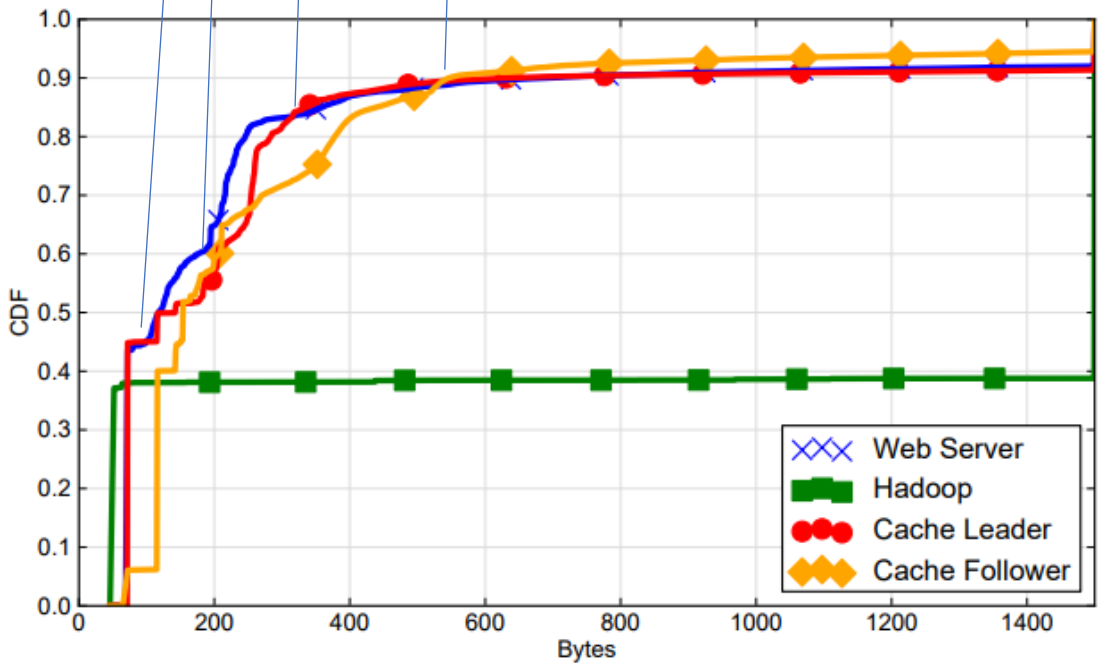
➔ Data from 1501 Files

The largest packets were removed to allow better plotting

Graph created from Trace Data:

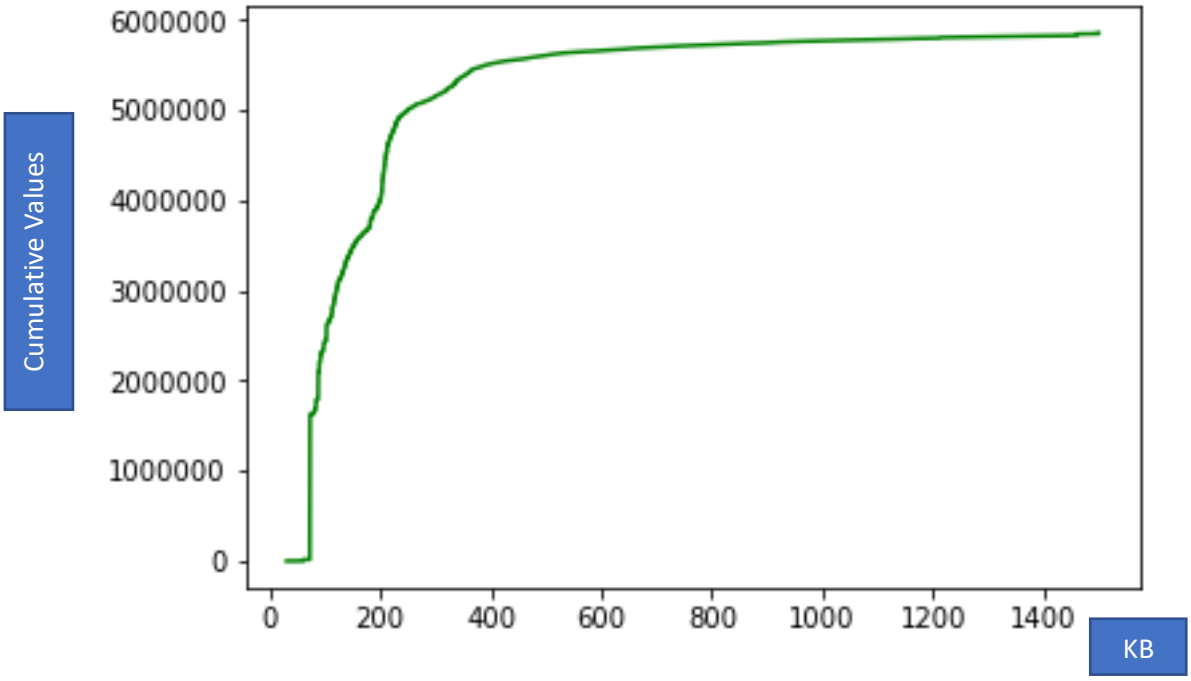


Graph taken from the Social Network Paper:



The arrows on the page are to mark the similarities in both traces.

The graph below is of the same data, from the same time, except it created with 400 seconds of trace data from the same hour, instead of just 120.



- ➔ Despite our data being heavily down-sampled, we're getting a similar spread of packets and their sizes, meaning -> good distribution for approximation.

Known Problems with This:

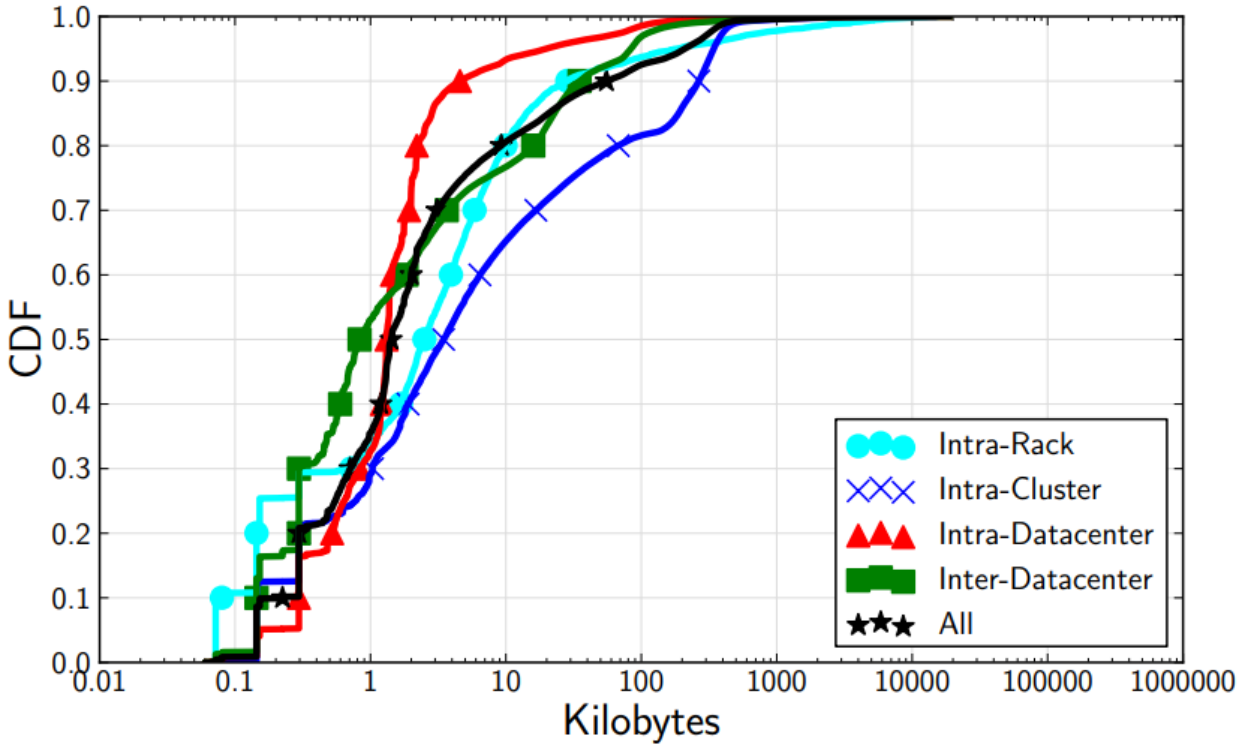
- ➔ We're not sure whether the data used in the paper to make their graph was from **Port mirroring** or whether they too were using **FBFlow Sampled data** to create it.
- ➔ Most of the trend itself is to do with the smaller packets, which are "usually" and probably mostly "syn" and "ack" and similar 'TCP protocol maintenance' packets those being self-contained single packet flows.

The issue with that, is that if FbFlow sampled and picked up a 72 byte Ack packet, that is the only packet of its own flow, meaning we got the entire flow. Everything seems good, but larger flows may still have missing packets. So while this graphs confirms we're ok when it comes to the smaller packets, meaning short flows have been somewhat preserved, long flows are still a question mark.

Positive Takeaways from this:

- ➔ **From the paper:** *"Packets for the other services have a much wider distribution, but the median packet size for all of them is significantly less than 200 bytes, with only 5–10% of the packets fully utilizing the MTU" =* meaning, perhaps we can work without the exact data for the largest packets and flows, which are less likely to be complete.
- ➔ Trend **is** being followed. If the packets in our sampled data are distributed the same was as in the actual data then there is a good chance that the scaling factor of the FBFlow scaled data will still maintain the basic intrinsic data characteristics.

Paper Graph 3: Flow Data Size Distribution



This Graph is where our own data starts to trouble us. Since we have no idea of flow information. Nor do we have sub-second time markings.

I am thus going to estimate flows from the same data being used for the other graphs features previously, using the following few 'parameters':

1. Using the standard 5 tuple flow separation [a flow is a unique set of 5 tuple values]
2. Filtering those flows by seconds. Only if any two traces, from the same src/dst IP and Ports are within the same second, are they the same flow.

This has the obvious advantage over 5 tuple flows over the entire data set, since if the same 5-tuple is present in two places in the data but they are say, 30 seconds apart. They are two different flows.

Data was broken into the 4 categories based on the separation criteria used in Zarq's own plotting code file:

1. IntraRack = [RackSrc == RackDst and InterDataCenterBool == 0 and InterClusterBool == 0]
2. IntraCluster = [RackSrc != RackDst and InterDataCenterBool == 0 and InterClusterBool == 0]
3. IntraDataCenter = [InterDataCenterBool == 0]
4. InterDataCenter = [InterDataCenterBool == 1]

I'm still not 100% on this since I'm not sure what the separation scheme should be. For instance, all **InterRack** and **IntraCluster** Traffic will be replicated in the **IntraDataCenter** category.

In my mind however, IntraDataCenter should be just the data that goes from one cluster to another. It should not be any traffic that stays within a rack because then it would just be rack local, or within the same cluster because then it should just be cluster local.

But this criterion is open to interpretation. I have used it here to keep things consistent with Zarq's own analysis.

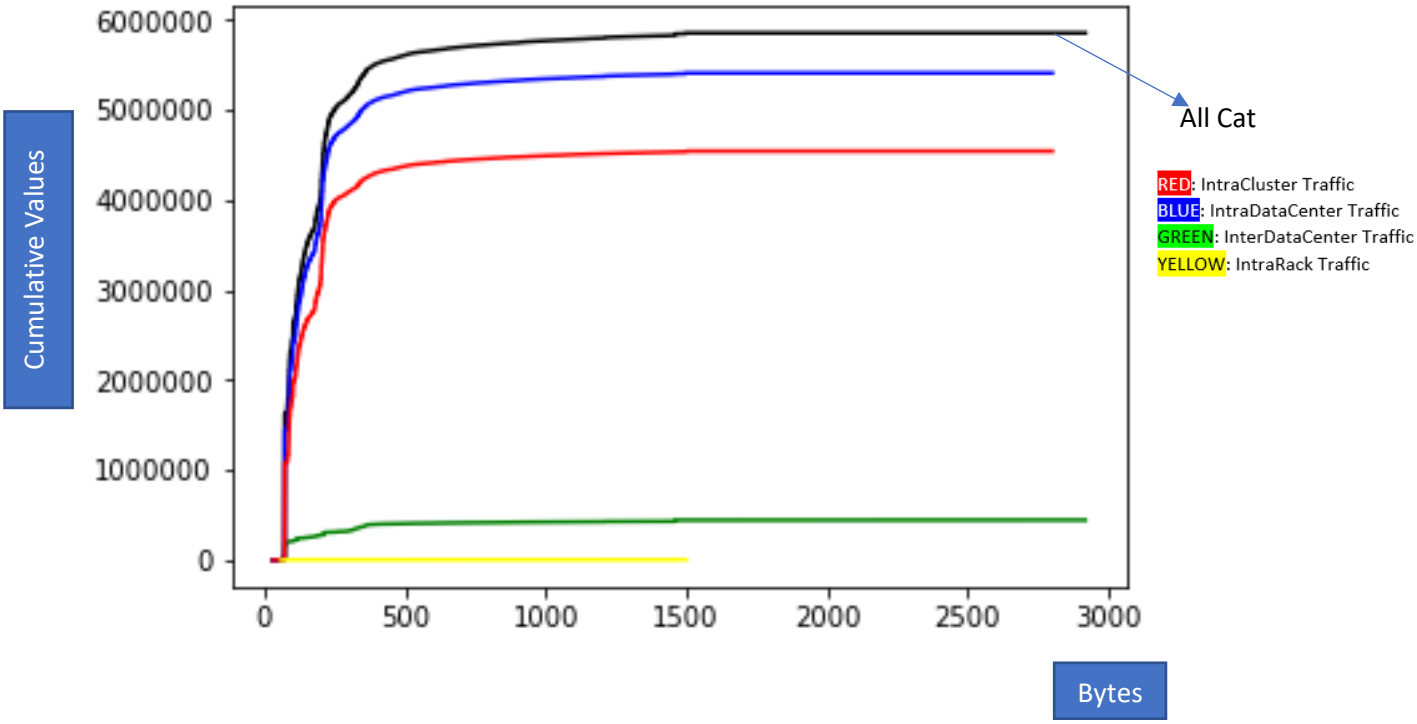
grpDataSum - Series

Index	1
(1475380800, '000a0098e73ab0d5', '4653d7a426285d1f', '8665175c', '7e4a071f')	0.000383
(1475380800, '000a0098e73ab0d5', 'fd82817ab3abc0ab', 'dd5703bd', '45d2b346')	0.000186
(1475380800, '000e6213967f0862', '528f70ce9c0cee46', 'bc382e43', '45d2b346')	7.2e-05
(1475380800, '000e6213967f0862', 'fe02f096585b4457', 'ff0006a5', '9374d216')	8e-05
(1475380800, '00183671fba80590', 'e14610226c948acc', '495d565e', '495d565e')	0.000112
(1475380800, '0023e00837954c35', '0fa918f5d2fb4bab', '45d2b346', '11d65909')	8e-05
(1475380800, '0023e00837954c35', '33f09f467ec783c4', '45d2b346', 'bb387168')	9.1e-05
(1475380800, '0023e00837954c35', '564e2c01e8d143e6', '45d2b346', 'caca6468')	8.8e-05
(1475380800, '0023e00837954c35', '5a4166d9e7711203', '45d2b346', 'd822d04c')	8.9e-05
(1475380800, '0023e00837954c35', '5f0c628c40a6e6b1', '45d2b346', '9c3a8918')	7.7e-05
(1475380800, '0023e00837954c35', '5fb32174fb689068', '45d2b346', '96de3c07')	0.000123
(1475380800, '0023e00837954c35', 'cad574fd3514c26f', '45d2b346', '7b826676')	8.9e-05
(1475380800, '0023e00837954c35', 'd868b5fe73711d07', '45d2b346', '6c77e403')	0.000528

The data looked like this, as a Pandas DataFrame. As can be visually confirmed, we have data sorted first by second, then by source and destination IP's and then Ports. All records that were the same have 6 tuple (including time) are now a "flow" and their packet sizes are put together and turned into a "flow data size"

Data was broken into the four categories and then "grouped" like this ^

Using Zarq's Criteria for breaking the data apart: The following Graph was created separating the various flows:

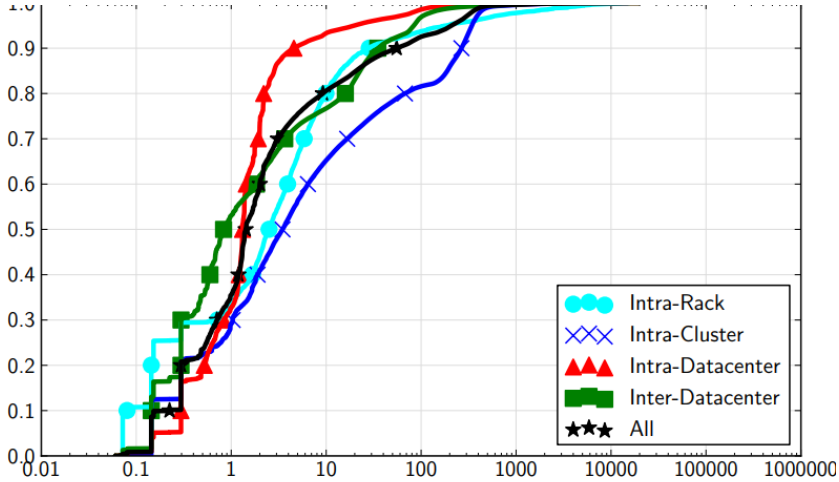


The same 400 seconds of trace data were used -> which resulted in 6 million plus trace records. Using about 5 million 850 thousand meant that the "skewed" data points were removed and the graph looked slightly more usable.

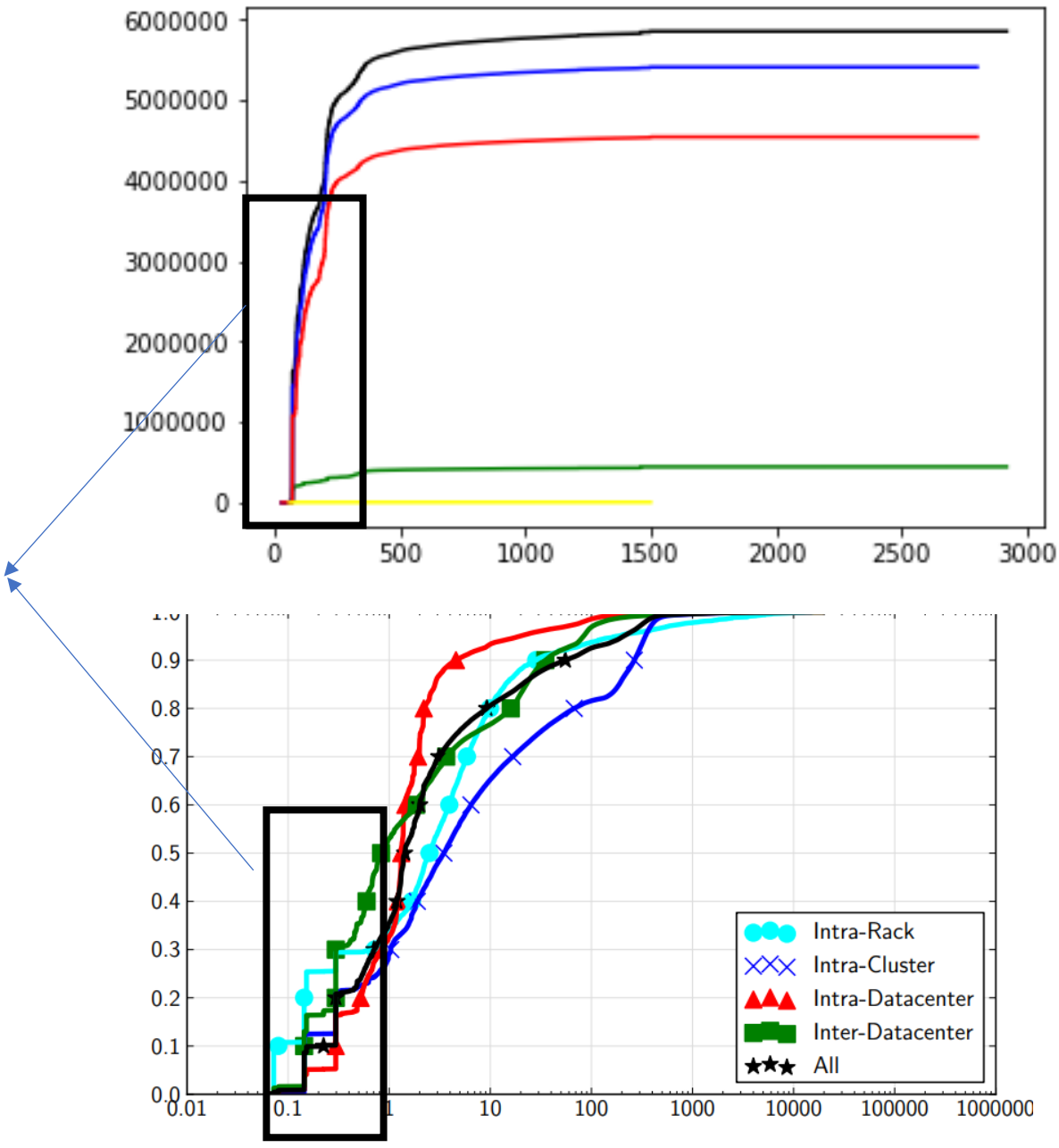
Notice that the scale here is in Bytes and not KiloBytes. This is the first sign of trouble. We can see we're missing a lot of interRack data here esp. Although since we're not sure what data was used to make the Facebook Graph we cannot blame the entirely different curve on the missing data.

Perhaps the data in this hour, despite showing good trend correlation in the previous graphs was distributed like this, even in the real world. Or perhaps when we break it into the different categories, we can see more clearly the issue with sampling. While overall, trends may be maintained, breaking data into categories loses it.

This is the actual graph reproduced here again:



But seeing here, some parts of the graph seem to agree with each other.



As identified, some trends are being maintained, for some categories. Not for all though, and not throughout the graph. The usefulness of this is questionable.

Conclusion:

Graph Types Considered:

1. Data Flow Cumulative Distribution
2. Packet Size Cumulative Distribution
3. Flow Size Cumulative Distribution

While Graph Types 1 and 2 gave good hope regarding the usefulness of the data. Graph Type 3 and its analysis show that, when going into the more granular time and categories, we do indeed start to lose out on some critical information that doesn't appear to be retrievable.

Again, bear in mind that these graphs are **ONLY for 400 seconds** worth of network activity, recorded in 1500 of the total 2700 files.

Perhaps with using all the files, and not expecting complete information and knowing it's limits good generalizations can still be made. Graph types 1 and 2 strongly support that. Graph type 3 shows more of the limitations.

But at this point, it is clearer how limited exactly. How bad the limitations are and where they show up more. As we keep zooming in to get more precise information, we will lose accuracy. If we split the data into the different categories we will definitely lose accuracy.

But keeping a "zoomed out" perspective can still provide a reasonable estimation of network activity at the time.

One thing is clear, using ALL of the data and as much of it, duration wise, as possible to draw conclusions is certainly a must, meaning some faster machines are going to be needed. Since we don't have the luxury of getting all the packets detailed, using all the ones we have is by this point, very clear to be imperative.

Another thing clearer, is that general trends (those not needed data to by split into 4 sub categories) will provide better generalization than split data will.

==========