A lightweight measurement platform for home Internet monitoring

Ian Marsh
SICS
ianm@sics.se

ABSTRACT
CheesePi is a lightweight platform written in Python with the goal of performing measurements from within users’ homes. By leveraging the Raspberry Pi multimedia content can be played directly to their HDMI display devices.

The real advantage of CheesePi however stems from networking the Pis and performing measurements in parallel to and from several locations. In order to ensure maximal usefulness of a networked measurement system, we enrolled the Swedish regulator into this project.

On this platform we have conducted home experiments on single-node configurations; TCP protocol performance, capacity measurements and VoIP experiments using two Pis. In a multi-Pi setting we have used the CheesePi network to monitor several large media events.

1. CHEESEPI PHILOSOPHY
We present a distributed measurement system for home Internet quality characterisation. The mission for CheesePi aims to objectively characterise the quality users obtain from their home Internet connections. CheesePi leverages the Raspberry Pi, an always-on, affordable, quiet measurement node acting independently of other devices in users homes [MM15].

We deemed home networking characterisation requires continuous monitoring of connections, an external device satisfies this requirement. Furthermore, in a home setting, non-technical requirements on the installation, configuration and presentation of data are important requirements. CheesePi can best be thought of as community software1 running on commodity hardware.

1.1 What CheesePi is not
We should point out like many of the works described in the related work, our objective is not to aim for a large scale deployment of measurement nodes. Rather, we want to deploy the minimum number of nodes to collate sufficient statistics for our regulator. Furthermore, we are aware that measurement efforts can generate large amounts of data, and wherever possible we would like to avoid processing issues by judicious choice of node placements and measurement metrics.

1.2 Home networking
By measuring from the home, we must consider a device that is quiet, relatively cheap, non-obtrusive, and be able to replay media. We opted for an external device, as we might need to measure a device that we could be using (a laptop for example). WiFi is important as we will need to move the device around and for passive measurements. Recording traffic volumes, not the contents is a requirement, as well as a pleasing interface, supporting different technical levels of home users. We tried other single card computers, in particular the Odroid device for the capacity measurements (Section 6.3).

1.3 Always on
Snapshot-like looks such as Speedtest or the more research oriented Fathom [DST +12] are precluded from our philosophy as they require users to click a browser. By being on we must make continuous measurements, minimal over weeks, and more probably over months. Continuous measurements, or continuously schedable, are not only important for an individual home, but for correlating with other devices. Measurements that rendezvous are essential to draw inference from, for example during poor quality streaming events.

1.4 Community software
The real usefulness of CheesePi stems from using several networked Pis simultaneously. Inference of Internet paths using a collaborative network yields information about shared Internet infrastructure over relevant timescales. Only by using a selected set of measuring nodes, can the correct user demographics and statistics be obtained for a regulator. Typical data collated includes: connection technology, number of inhabitants/home, age distribution, geographic

---

1available on the global PyPi Python repository.
region, ISP provider. As a community project, Python is an important choice, to build a platform quickly, use existing libraries, and draw help on engineers using current programming languages. A small amount of C was needed for the networking tools.

1.5 Unbiased, neutral measurements

Our central tenant is a representative unbiased neutral sample of home connection quality within Sweden. We wanted to ensure, that not only are individual measurement points constrained to a budget, or share thereof, but also each ISP receives a fair share of the measurement budget. This is really at the behest of the regulator for obvious reasons of neutrality. Table 1 shows some options for how to perform active measurements over several ISPs or operators.

1.6 Country specific

Although not constrained by the software per se, we expect CheesePi to be used within a country. This is because some contact with the regulator is useful within a country. We have hinted at a deployment of measurement nodes, to be inline with regulator needs.

For example, a deployment should include an appropriate proportion of DSL/fiber connections, rural/urban, regions and age demographics and so on. We give some examples in Section 9. Therefore country specificity is easier. Also, some knowledge of the ISPs can help, including personal contacts. In one case, an ISP needed rates generated that we had not dealt with, so needed to look at the capacity issue, see Section 6.3. CheesePi has some focus on popular media events (see section 7) and these might be slightly different in each country or region.

Importantly, SICS is non-profit organization with no financial nor competition incentives, therefore is well suited to undertake a measurement task together with the regulator.

1.7 Multimedia

From a multimedia perspective, the Pi has an HDMI interface capable of displaying H.264 video with 6 channel Dolby audio. Indeed, one of the many uses of the Pi in a home is as a media player using the KODI dashboard. We measure the quantities necessary to derive QoE-like metrics.

For example, in order to quantify the frustration associated with video stalls, we measure the current video rate, coding format, available capacity, receiver buffer lengths plus a few others to collate the most important factors. The goal is to link measurable quantities to the frequency of play-stalls, however we stop short of producing QoE reports. We want to provide data to the QoE community, but do not include users in our loop to completely automate the process.

2. RESEARCH AND THE REGULATOR

This work is presented in collaboration with the Swedish regulator, the Post and Telecom Authority (www.pts.se). Within this project the regulator is concerned with expanding home Internet connection quality tests from naive throughput and delay measures (a la Speedtest), into meaningful data collated over intervals. The regulator’s influence on this work extends the community approach, where the community is the demography of users needed to sample all users in a nation and fair measurements to maintain a balance a regulator must observe.

At the EU level, the Body of European Regulators for Electronic Communications, BEREC, has the goal of providing fairer services for European users. BEREC makes recommendations primarily towards ISPs and operators for QoS within a net neutrality setting. Within CheesePi, service and quality monitoring of connections is paramount and been guided to some degree by BEREC’s requirements (oEC/CE14).

3. RELATED WORK

Internet Measurement platforms are certainly not new, [BS15] describes many efforts up to 2015, an updated publication from the same authors is in [BBE +16]. A text on Internet measurements by Crovella and Krishnamurthy from 2006 covers many areas still relevant to measurements today [CK06].

The Archipelago Measurement Infrastructure from UCSD uses 150 PIs, globally deployed to quantify connectivity and performance using IPv4 and IPv6 active measurements organized as an overlay [AR]. Quantifying the congestion is done by a Time-Sequence Ping, a crafted sequence of pings. Commercial PI-based offerings include NetBeez and NetRound.

RIPE Atlas is a project measuring Internet quality around the world using active probes [BES15]. Around 9400 probes have been deployed as of 2017. ICMP and traceroute measurements are performed using the hardware probes with security support by DNSSEC and SSLCert. The goal is to monitor critical DNS servers and measure the latency between most test nodes, they collate about 6 PB data / year.

SamKnows is a project that measures broadband performance for consumers, ISPs, and regulators [Sam]. The Whiteboxes are connected to the router and run down/upload throughput, VoIP quality, FTP throughput, DNS resolution latency, P2P throughput, web page load latency, email latency and several video QoE tests. Samknows also measure cellular quality via their smartphone apps.

BISmark is a testbed that utilises modified home routers in order to perform both experiments and measurements [SDFdD14]. The routers perform active measurements but are also capable of passive measurements with user consent. A few hundred BISmark routers have been deployed by the end of 2016, however BISmark is generally opposed to external changes to their system, quite different to our community approach.

Teacup is a joint project between Swinburne university...
in Australia and Cisco, with 100s of deployed nodes using PCs written in python [ZA15]. The focus of their research is the TCP protocol, as we do in the dual mode operation of CheesePi.

Dasu is a software-based tool for end measurement characterisation of the ISPs performance [SOB+13]. Their goal is to push experiments to the Internet’s edge, allowing a smaller scale measurement approach. Our project is influenced by one of their ideas of a network budget for the total amount of data to be sent (by the community) and consequently implications for each home measurement. We take this one step further, by making fairness as key component in a measurement approach.

What exactly fair share means is not well-agreed up within the measurement community. Fair sampling has been touched up [Duf12], but fair measurements, as far as we know have not been researched nor mandated by regulators.

4. CHEESEPI DESIGN

The CheesePi platform is designed to be modular, simple and lightweight. Such requirements led to Python, with support of its libraries e.g. Twisted, txmsgpackrpc and Scipy/scikit-learn. CheesePi is essentially six components:

1. A key-value local database
2. An SQL centralised database
3. A measurement dispatcher
4. Measurement tasks
5. Presentation dashboards
6. Data processing tools

They are a loosely coupled set of modules as illustrated in Figure 1.

### 4.1 Database storage

The client database stores measurement data from the Pi it resides on. The data is stored in a structured format inside a schemaless database engine. The advantage of a schemaless storage engine is that custom built measurement tasks can easily store their output without requiring restructuring of the database. Influx is the currently supported lightweight database. It executes without on a Pi with acceptable performance. It provides good support for time-series data, MongoDB is also available as a client database. Both store the measurement data as JSON for human readability, and for relatively syncing to the centralised database. The central database is SQL-based.

Collated measurements can then be queried in a centralised database. Clearly one ISP should not be able to query others data. The centralised database SQL tables hold

- 1) administrative fields, IDs, times-dates and management/access data
- 2) raw measurement data, as well as processed measurement data
- 3) Pointers to raw data files whence processed.

Two examples of SQL commands are given, the next is the last 10 ping data from Pi with ID 3.

```sql
SELECT * from Ping WHERE PID=3 ORDER BY OID DESC LIMIT 10;
```

And a second example where we select the Pis using WiFi.

```sql
SELECT PI.PID FROM PI, Connection
WHERE PI.PID=Connection.PID
AND PI.EthMAC=Connection.MAC;
```

### 4.2 Measurement dispatcher

The measurement dispatcher runs tasks determined by a configuration file and on completion stores results in that...
Pi's database. Tasks can be set one of two priorities and can be scheduled asynchronously. For dispatch, each PI contains a universally unique ID, called a UUID, an 128 bit-OSF standardised value in hex format, e.g. 54a7969d-8436-49a3-a242-937dcf7c2d36. An important part of the dispatcher is a schedule, which resembles a crontab, however a little more user friendly, and which performs collisions in measurement tasks. Currently it does not implement the fairness properties discussed in the previous section, but will do in the near future.

4.3 Measurement tasks

A class template for writing measurement tasks is provided (≈100 lines) with 15 being available from WiFi polling to iperf suites, a simple (dummy) example is shown below. We expect developers to contribute tasks as needed.

```python
import time
import os
import cheesepi as cp
import Task

logger = cp.config.get_logger(__name__)

class Dummy(Task.Task):
    
    # construct the process and perform pre-work
    def __init__(self, dao, spec):
        Task.Task.__init__(self, dao, spec)
        self.spec['taskname'] = "dummy"
        if not 'message' in self.spec: self.spec['message'] = "test"

    # actually perform the measurements, no arguments required
    def run(self):
        logger.info("Dummy: %s @ %f, PID: %d" % (self.spec['message'], time.time(), os.getpid()))

# main call if __name__ == "__main__":
    dao = cp.config.get_dao()
    spec = {}
    dummy_task = Dummy(dao, spec)
    dummy_task.run()
```

5. SINGLE ENDPOINT CONTROL

A Javascript front end called grafana handles plotting in the standard install case, however other visualizations are available, as we want non-technical representations of Internet quality to be available. These are shown in more detail in section 5.

5.1 Technical dashboard

Where home users are using the Pi to monitor their home connections, we display the results for tasks installed and scheduled on a dashboard. Since the Pi has an HDMI interface, it can be connected to an external device for continual display. Figure 4 shows a grafana representation of the tasks we setup for this experiment. Row-by-row we show the ICMP delay, the HTTP delay to a Alexa-selected site, a speedtest to the nearest Ookla site, a YouTube retrieval of http://www.youtube.com/watch?v=_OBlgSz8sSM “Charlie bit my finger - again!”, the packet loss, the DNS lookup, the WiFi access points seen and some local Pi information, temperature, load and uptime.

5.2 Non-technical dashboard

Since we envisage CheesePi being used in a variety of homes, we designed some different designs for the dashboard. Figure 5 shows a Javascript representation of the download speed and latency “flowing” into a home. It is not designed to show the continuous nature of data flow. The mean rates are represented by the height and swell represents the variance of the traffic flow. Since network conditions tend to vary on small time scales (particularly wireless) we smooth the data, 80% old values and 20% recent values to make the visualisation more pleasing. Just in case the user is not completely sure of the rates, we added mouse-over events to show extra information, but are not shown by default. This also gives us the dual feature of a visualisation, natural by nature, but with a mouse, extra technical information can be made visible.

6. DUAL ENDPOINT CONTROL

Next we describe three scenarios that needed control over...
both endpoints such as in a client-server experiment, VoIP quality experiments and capacity tests.

6.1 In-protocol delay measurements

Client-server handshake: A simple application to measure the network delay using both ends of a TCP connection as shown in Figure 7. The idea is to use TCP as a “better ping”. An insight is to separate the protocol RTTs from the object retrieval. During a web page request and retrieval, the latency encountered is caused by a combination of the end-terminal and network processing. Since a normal client does not see the server side.

We use the CheesePi platform to investigate the delays associated with a client-server interaction. The full methodology is described in [Por16]. Other methods include where the authors attempt to predict the object transfer time using the delay estimate of the handshake through a model and cellular transfers. They are interested in ascertain how many web objects can be transferred in the slow start phase. We are concerned with more accurate delay measurements using the handshake-object retrieval-teardown phases to obtain the stability of the network delay.

Figure 8 shows a data set of 3000 ICMP requests at the same time as TCP handshakes. In this experiment, we included the end host processing time, as seen from the client. Although the processing done by the server between an incoming SYN packet and an outgoing SYN ACK packet is small, it is still significantly influences the resulting round trip time.

6.2 VoIP quality in the CheesePi network

The E-model [IT14] is a transmission planning model developed by the International Telecommunication Union. It takes 23 parameters as input and outputs the R-value, a numerical value describing the expected quality of VoIP calls. The model relies on an additive formula, where each part describes different quality aspects of a call.

\[
R = R_\text{o} - I_s - I_d - I_{e-eff} + A
\]

\[
R_\text{o} = \text{noise sources such as circuit and room noise}
\]

\[
I_s = \text{voice impairment to the signal}
\]

\[
I_d = \text{delay and equipment impairment}
\]

\[
I_{e-eff} = \text{packet loss impairment}
\]

\[
A = \text{advantage factor, potential beneficial factors}
\]

Table 2: Relation between the R-value, MOS and user satisfaction.

<table>
<thead>
<tr>
<th>R-value</th>
<th>Mean Opinion Score</th>
<th>User satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>4.34</td>
<td>Very satisfied</td>
</tr>
<tr>
<td>80</td>
<td>4.03</td>
<td>Satisfied</td>
</tr>
<tr>
<td>70</td>
<td>3.60</td>
<td>Some users dissatisfied</td>
</tr>
<tr>
<td>60</td>
<td>3.10</td>
<td>Many users dissatisfied</td>
</tr>
<tr>
<td>50</td>
<td>2.58</td>
<td>Nearly all users dissatisfied</td>
</tr>
</tbody>
</table>

Two of the parameters for the E-model can be gathered by RTCP reports: RTT and packet loss probability. With the rest of the parameters set at default values the R-value can be calculated. A module capable of generating the R-value was written in Python. It takes two parameters, RTT and packet loss probability. Some pairs experienced mean R-value and standard deviation that were very similar. This implies that there is little to no asymmetric behaviour between the two measurement nodes.

The mean and standard deviation are quite similar between two nodes in Stockholm and a node in Karlstad (half way between Stockholm and Oslo, about 300km), figure 9.

6.3 Available bandwidth measurements
Generating traffic to characterize large capacity network links with high accuracy in transport backbones is important for Internet service providers. We use the iperf3 network application to measure throughput, and indirectly congestion on links within a network. Using short bursts at high utilization can ascertain whether a connection can support capacity sensitive applications, such as video streaming. Repeating the process to capture day-night effects can categorize links for management decisions over busy-quiet periods.

From a request from 1 ISP, they needed CheesePi nodes to generate higher data rates, we looked at single-board computers other than the Raspberry Pi. These are summarized in the Table 3 below.

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi 1 (B)</td>
<td>Test machine</td>
<td>54.2</td>
<td>69.3</td>
</tr>
<tr>
<td>Test machine</td>
<td>Raspberry Pi 1 (B)</td>
<td>84.1</td>
<td>60.0</td>
</tr>
<tr>
<td>Raspberry Pi 2 (B)</td>
<td>Test machine</td>
<td>94.1</td>
<td>96.7</td>
</tr>
<tr>
<td>Test machine</td>
<td>Raspberry Pi 2 (B)</td>
<td>94.1</td>
<td>95.7</td>
</tr>
<tr>
<td>ODROID-XU4</td>
<td>Test machine</td>
<td>837</td>
<td>764</td>
</tr>
<tr>
<td>Test machine</td>
<td>ODROID-XU4</td>
<td>785</td>
<td>278</td>
</tr>
</tbody>
</table>

Table 3: Maximum throughput between test machines and measurement nodes (Bits/sec).

Using both ends of a connection is the mode in this section, iperf is one such application. The CheesePi platform and network allowed this to be done relatively simply and the delay between 2 selected nodes are shown in Figure 10. The round trip time distributions indicate good stability of the CheesePi network in terms of RTT, even when loaded with iperf traffic.

In the throughput case Figure 11 shows the achievable throughput between 2 Raspberry Pi nodes, which is at around 80% of the raw speed of the device itself. We acknowledge that sending full rate traffic into a network is somewhat disruptive, therefore wanted to ascertain the minimum time this needs to be done to obtain a reasonable estimate. From our discussions with ISPs, they routinely load the network for testing and diagnosis and have sophisticated (and expensive) tools to load their internal networks.

Full details of the work on accurate traffic generation, deployment in real networks and its use in CheesePi can be found in [Sha17].

7. MULTI-ENDPOINT CONTROL

In the third mode the Pi acted as a home node to quantify Internet quality during a popular event. One such event was a boxing match between Mayweather-Pacquiao in 2015. We gathered 10 hours of data resulting in 80K measurements from 3 distributed Pis to 3 servers screening the event. IP networking delays using ping and the HTTP frontend server responses using httping were captured. The latter tool measures a GET request to the remote webserver.
Table 4: Media server characteristics for the most televised Boxing event in history.

<table>
<thead>
<tr>
<th>Service provider</th>
<th>Philstar (Phil)</th>
<th>Showtime (Sho)</th>
<th>Sky (Sky)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>philstar.com</td>
<td>sho.com</td>
<td>sky.com</td>
</tr>
<tr>
<td>Role</td>
<td>News &amp; entertain. portal</td>
<td>US cable entertain. network</td>
<td>UK-based entertain. network</td>
</tr>
<tr>
<td>Server location</td>
<td>Arizona, USA</td>
<td>Ams, NL</td>
<td>Ams, NL</td>
</tr>
<tr>
<td>CDN provider</td>
<td>-</td>
<td>Akamai</td>
<td>Akamai</td>
</tr>
<tr>
<td>Timezone difference (hrs)</td>
<td>-7</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>Internet hops</td>
<td>12</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>ICMP delay (ms)</td>
<td>126.5±36</td>
<td>29.5±2</td>
<td>2.2±2</td>
</tr>
<tr>
<td>HTTP delay (ms)</td>
<td>418±128</td>
<td>79±130</td>
<td>24±62</td>
</tr>
</tbody>
</table>

Figure 13: HTTP latencies from CheesePi nodes to 3 global media servers hosting Mayweather/Pacquiao boxing event.

and mtr were used as reachability tools.

Figure 13 depicts the HTTP delay to our chosen three sites. Clearly there is an increase in the delays towards the event. The total response times increased up to and including the match itself. One question is can we separate out the difference between the network and server delays? It is possible that either could be the cause of the observed delay. For the Philstar connection, in terms of delay, we refer to Figure 14.

Clearly the average delay is due to the network, however the variance in the delay is due to the server response. To quantify the contribution of the network and server a principal component analysis can be done with both components being positive and the component due to the server delay being around $4 \times$ that of the network delay. Analysis of the data gathered by the nodes is an integral part of our philosophy.

8. DECOUPLING NETWORK AND SERVER DELAYS

Problem: We would like to understand the impact of delay, which is a function of the network, server and end system performance. Since they are coupled systems, the interaction of the network and server delays can be difficult.

As we will achieve this by deconstructing the correlation and showing the effect of the separation by clustering techniques.

![Figure 14: Network and Server delays extracted from figure 13.](image)

we can assess the impact of the individual systems on the total delay seen by the streaming video (in this case).

Therefore we evaluated two unsupervised learning methods to classify the delays on existing data sets. The methods considered are Principal Component Analysis (PCA) and t-distributed stochastic neighborhood embedding (t-SNE). Subsections 8.1 and 8.2 describe PCA and t-SNE that achieves a clustering of networked data with a 2D representation. A review of dimensionality reduction, by the author of t-SNE is given in [vdMPvdH08] whilst a highly cited book on PCA is [Jol86].

8.1 Principal Component Analysis (PCA)

PCA is a variance-based method for reducing the dimensionality of datasets. It uses an orthogonal transformation to convert a dataset of correlated variables, into a set of sorted values of linearly uncorrelated variables, called the principal components [Pea01]. The transformation ensures the first principal component has the largest variance, the second the next highest variance, and so on. The number of principal components will be less than or equal to the number of original variables, hence one can reduce the dimensionality by exploiting that some of the variables contribute little to the dataset and can be ignored. Furthermore, one can visualize the relative importance of each component with the data.

8.2 The t-distributed stochastic neighborhood embedding algorithm (t-SNE)

t-SNE is a non-linear machine learning algorithm for clustering high-dimensional data by projecting each measurement point onto a 2D scatter plot [VDM14]. On a low dimension visualization, the similarities, via clustering become visible. The algorithm constructs a Gaussian distribution over pairs of measurements in the high dimensional space (just delay in this case $\mathbb{R}$, or the measurement space generally. A Gaussian is chosen to group similar points close to the center point. Pairwise points with similar measurement values have a high probability of being picked by a conditional term in t-SNE. The variance, or range is extracted from the data. Another distribution is created in
the low dimensional map, as shown in Figures 17 and 18, but with a slightly heavier tail, based on a student-t distribution. Points are moved in the LD space to minimize the difference between the HD and LD spaces, whilst still producing pleasing results. Stochastic gradient descent and the Kullback-Leibler distance are used to measure the “difference” between the LD and HD maps.

8.3 Delay distributions for site quantification

Delay is an important QoE metric when watching online video, as in the boxing match. Delay impacts video streams buffering for an individual user and unsynchronized viewing across multiple users. For three sites, Philstar, Showtime and Sky we show the ICMP and HTTP delay distributions as kernel density estimates in Figures 15 and 16. From the former we can see that the delay distributions overlap, making separation, with or without times-series information difficult. It is clear that the variance of the HTTP delays are more variable, seen by wider distributions in the figures.

![Continuous server (HTTP) delay distributions of the boxing match per operator.](image1)

Figure 15: Continuous server (HTTP) delay distributions of the boxing match per operator.

![Continuous network (ICMP) delay distributions of the boxing match per operator.](image2)

Figure 16: Continuous network (ICMP) delay distributions of the boxing match per operator.

8.4 PCA and t-SNE as classifiers

Figures 17 and 18 show the results of the analysis presented in the previous section. We used only the HTTP measurements in this classification work, with one measurement being represented as one point (a text label) in the figures. The two components are shown on the x and y axes.

The initial results indicate that the delay distributions from the server responses are sufficient to, in most cases, to uniquely identify the streaming servers. A small percentage ~1.5% of the Showtime measurements were grouped far from the main components in PCA and as Sky and Philstar in t-SNE. We attribute these to some high delays between 16:00 and 18:00. One can see that t-SNE’s placement is somewhat better than PCA, which it is designed for.

Direct comparisons should be treated with some care. The distance metrics for PCA and t-SNE are not the same, as PCA uses a Euclidean space, whereas t-SNE uses the Euclidean in the HD space, but not in the LD one. This can be seen as the different ranges and values of the component values in the two plots. One can observe some data points in t-SNE which potentially are miss-classifications. Further analysis of each point is needed, and it is possible they are identical in delay values at these times, which would make them indistinguishable for one factor.

One important difference not seen from the plots is the running time, for our 40K measurements, the PCA classification takes around 1 minute whereas the t-SNE, with a Barnes-Hut optimization (better layout), takes about 2 minutes on a 2015 Macbook Pro. As an offline technique this may be acceptable, as an online analysis might require further optimization.

![PCA clustering and visualization of the HTTP server delays of the boxing match.](image3)

Figure 17: PCA clustering and visualization of the HTTP server delays of the boxing match.

PCA is parameter free given the data and only requires the principal components. t-SNE relies on some parameterization, in particular, how the some parameters produce different looking visualisation. Large data sets are almost inevitable in network measurements. Representing the whole structure with local detail is a challenge. t-SNE can be used to visualize large datasets, by performing random walks on neighborhood graphs, which allows local structure as well as a complete view of the data.
9. HOME MONITORING USEFULNESS

Writing technical applications is relatively straightforward, however working with real users on their daily habits is another discipline entirely. Domains of human psychology and behavioral studies are needed to ask and interpret the relevant questions, which we have done in the context of home Internet use, problems and whether something like CheesePi would be useful. The following extracts from our survey show what kind of users we had, how they connect, whether they experience problems and if CheesePi could be useful.

The full report is at https://cheesepi.sics.se/survey.pdf.

10. FUTURE WORK

The future of the platform will continue in several directions.

1. The roll out of Pis, we are looking at around 100 nodes in Sweden to cover the demographics and statistics required by the regulator. An open question is how to conduct the measurements in an unbiased manner; should each ISP receive the same number of measurements? or, in proportion with the number of customers? We mentioned this in the first section, but some additional research is needed.

2. One goal of the platform is obtain a better understanding of the root causes of buffering video events. Are the “rings of death” due to network capacity, full queues, poor coding choice, cross traffic and so on. The machine learning work is ongoing with extending the classification-clustering, by more factors than just delay. In order to obtain which factors contribute to buffer stalls for example.

3. Detecting if separate end-systems share a common path to a server can explain poor correlated spatial and temporal congestion. Where are traceroute cannot reveal topology [CCW14], inference at the end system poses an interesting research challenge [RKT02]. This will be developed with an operator in Sweden.

4. From an implementation point of view, we are working on NAT traversal, and are up to about 55% pene-
11. CONCLUSIONS

In this paper we have described the philosophy, design, implementation, data, analysis, actors, user feedback, future and some conclusions from our multimedia-oriented measurement platform. It is the culmination of around 1 years work and produced four Master’s students.

We have conducted many experiments using CheesePi: Voice over IP, capacity planning, popular media events, node similarities to name just a few. We have well over half a million measurements from several popular media events hosted on the Internet. The regulator’s knowledge of the broadband population will be leveraged in the placement of CheesePi nodes. This will ensure sound statistical coverage of the operators access technologies and demographics. Further rollout is ongoing in 2017, even to other nodes with the Nordic countries.

The software is free to download, with instructions and links to the existing work at cheesepi.sics.se. The project’s site has a link to the public code repository, gathered data, user survey and live demonstration of the dashboard. We have measured the delays, both network and web server, and the network path to four major media events via Ethernet and WiFi links, to streaming server front-ends.

We have compared variation analysis technique, PCA, with a relatively novel technique, t-SNE, found in the image analysis and classification communities for our datasets. We need tools such as these for our popular media events taken from CheesePi. Some of the higher network and server delays resulted in poor user quality manifested as video buffer stalls during the boxing event in this case. We also performed measurements during the Eurovision song contest and Champions league finals.

We believe that network researchers, regulators and designers should collaborate to improve the understanding of digital service dynamics. With some development behind us, we are looking to grow and develop the project given experience from the existing measurement community.

12. ACKNOWLEDGMENT

This work was financed by the Swedish Knowledge Foundation project entitled “Research environment for low-delay Internet”, under grant number 20130086.

13. REFERENCES


[SFBfD14] Srikanth Sundaresan, Sam Burnett, Nick


