Pedro Casas

Telecommunications Research Center Vienna – FTW

Traffic Classification in Networking

Applications to Traffic Monitoring and Analysis

IIE – FING – ARTES

1–5 September 2014

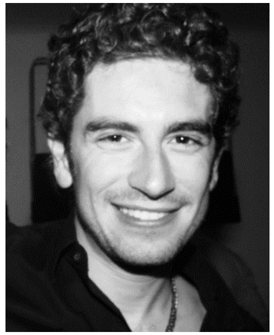
Thanks giving to many colleagues

- *The material presented in these slides is partiallty taken from the work done by Prof. Marco Mellia @Politecnico di Torino*


Marco Mellia
Politecnico di Torino


Raimund Schatz
FTW


Arian Bär
FTW


Pierdomenico Fiadino
FTW


Ernst Biersack
EURECOM


Alessandro D'Alconzo
FTW


Tobias Hossfeld
Würzburg Universoty


Mirko Schiavone
FTW

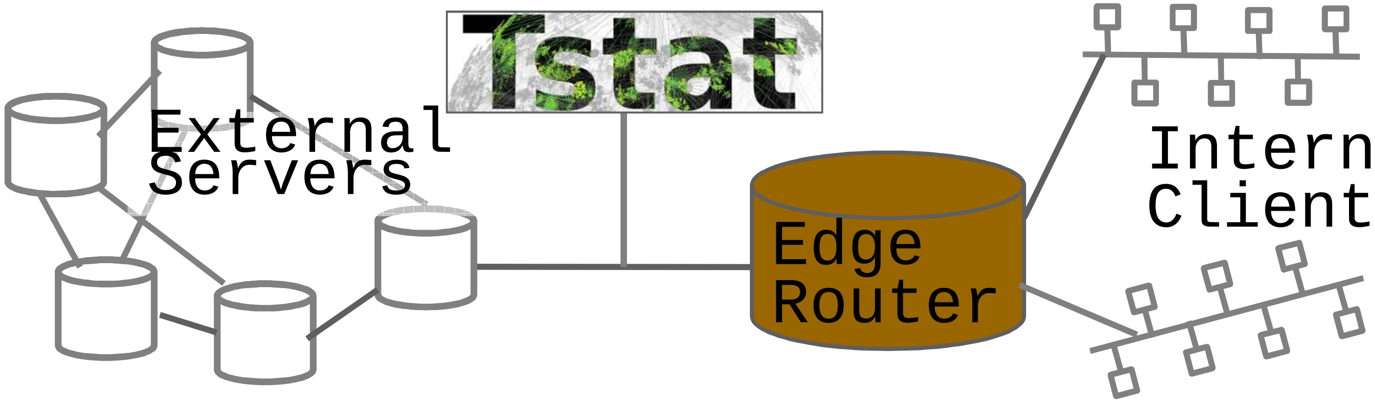

Philippe Owezarski
CNRS


Alessandro Finamore
Politecnico di Torino

Traffic Classification & Measurement

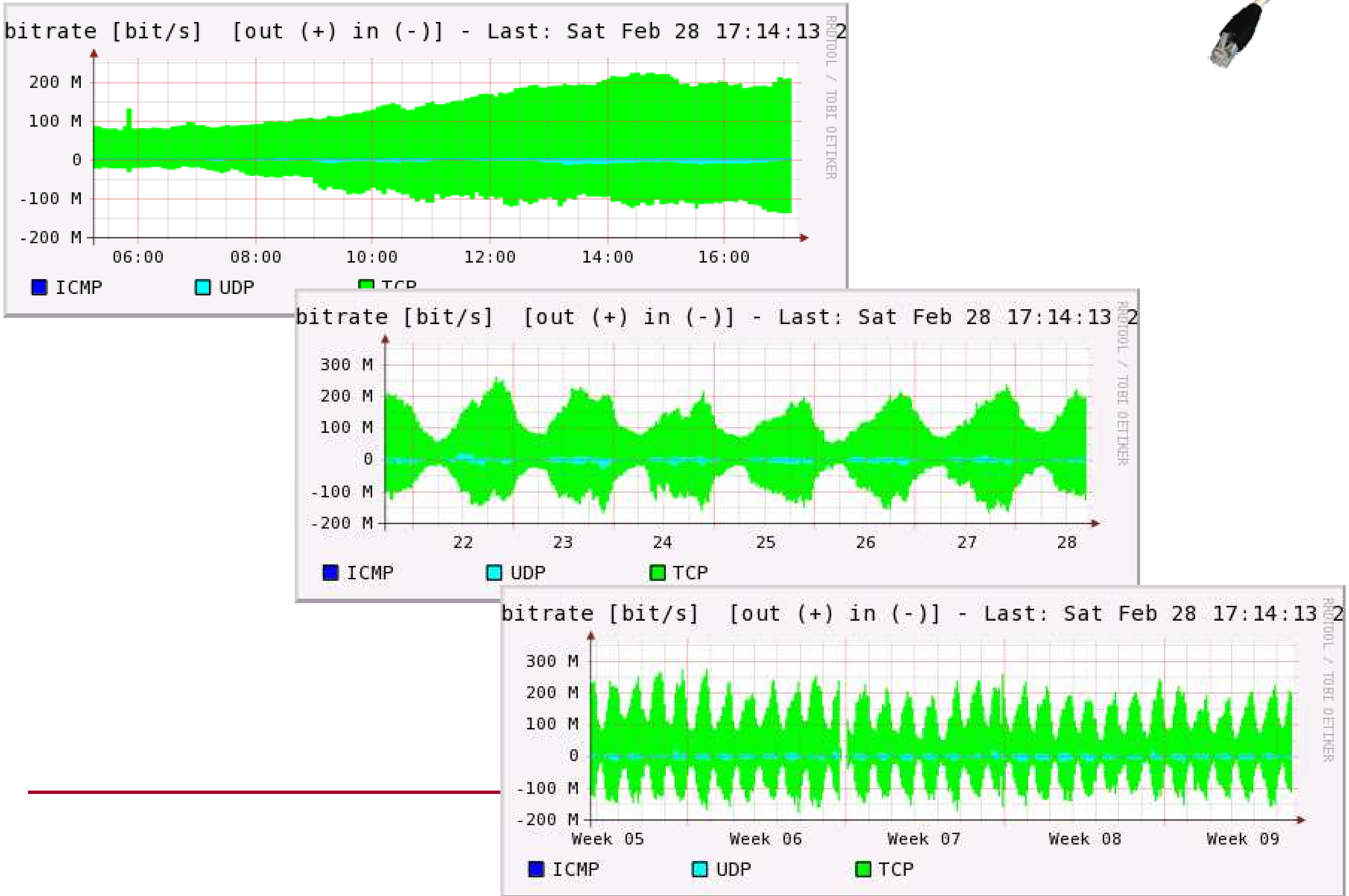
- **Why?**
  - Identify normal and anomalous behavior
  - Characterize the network and its users
  - Quality of service
  - Filtering
  - …
- **How?**
  - By means of passive measurement

Scenario

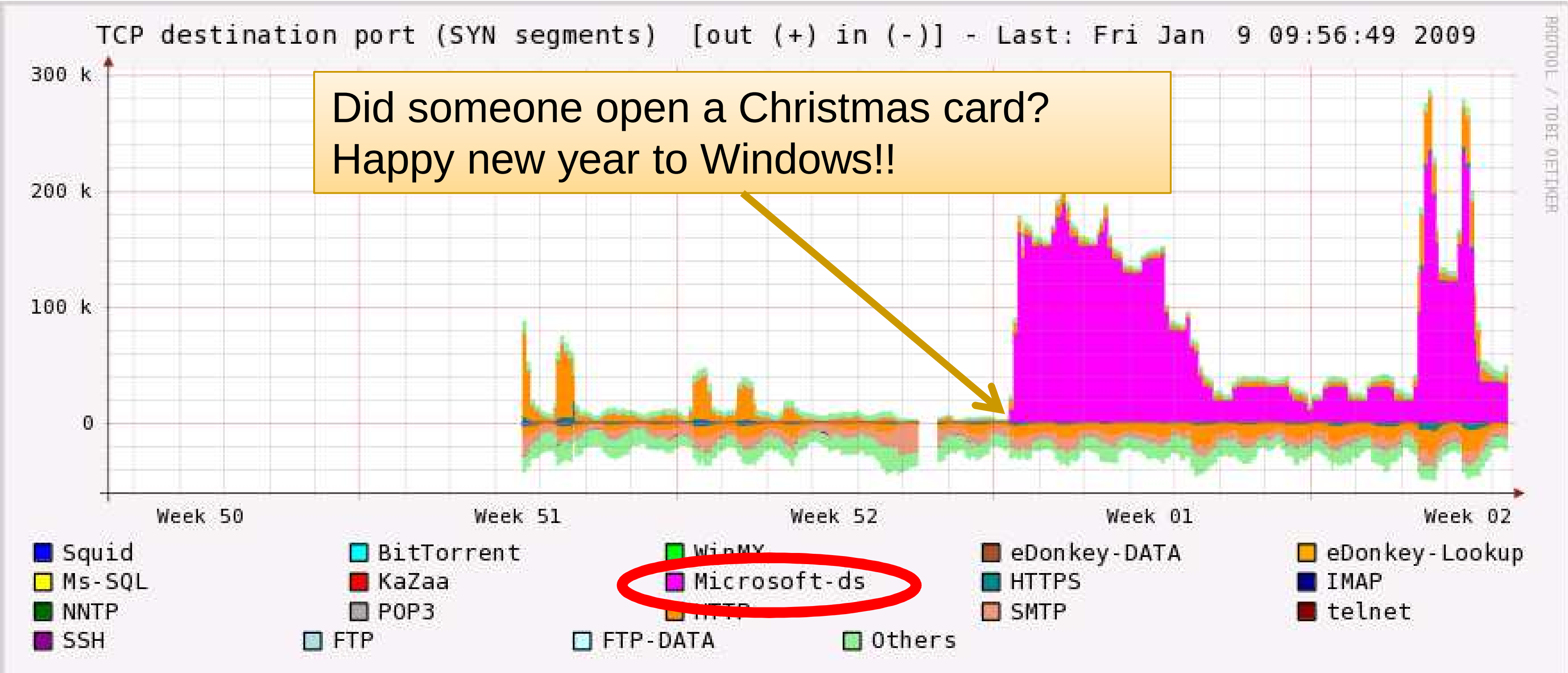


- Traffic classifier
  - Deep packet inspection
  - Statistical methods

Traffic Views from Tstat

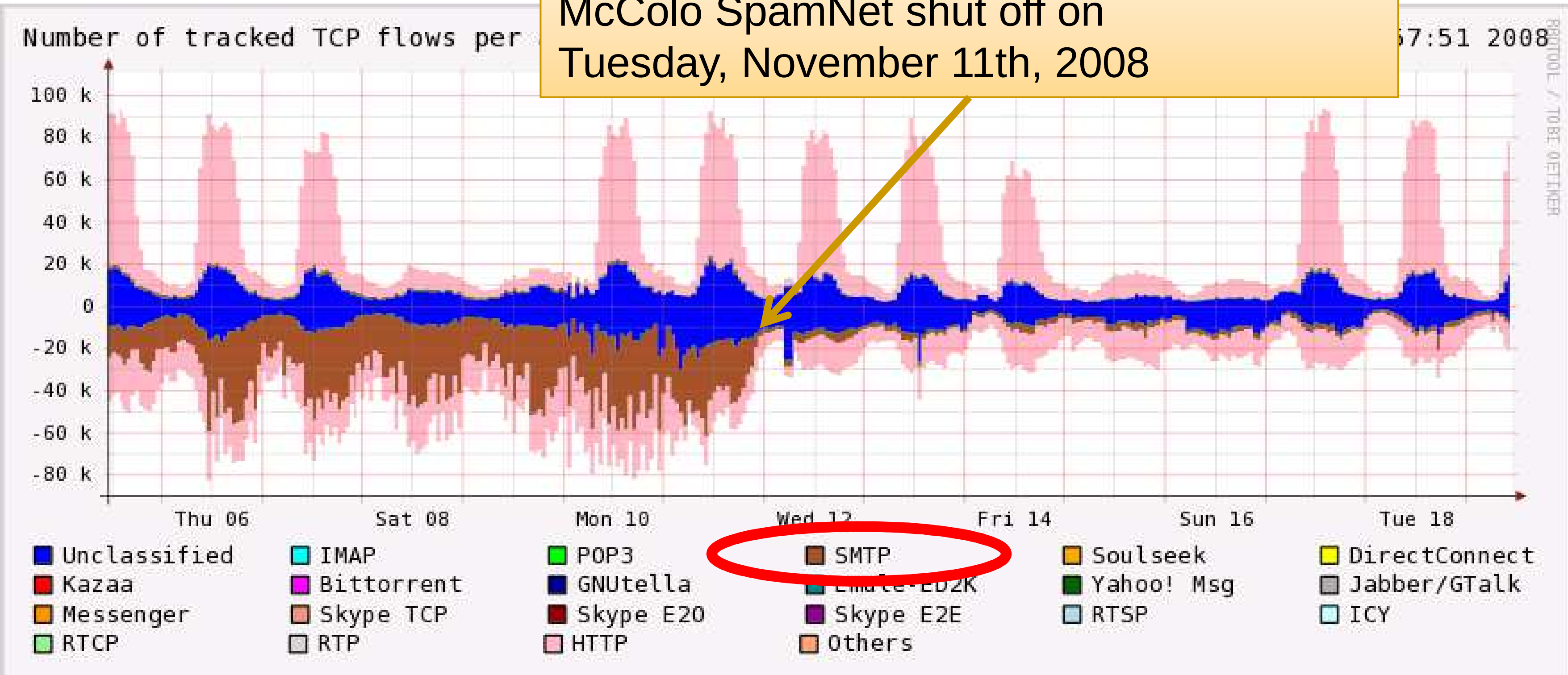


Worm and Viruses?

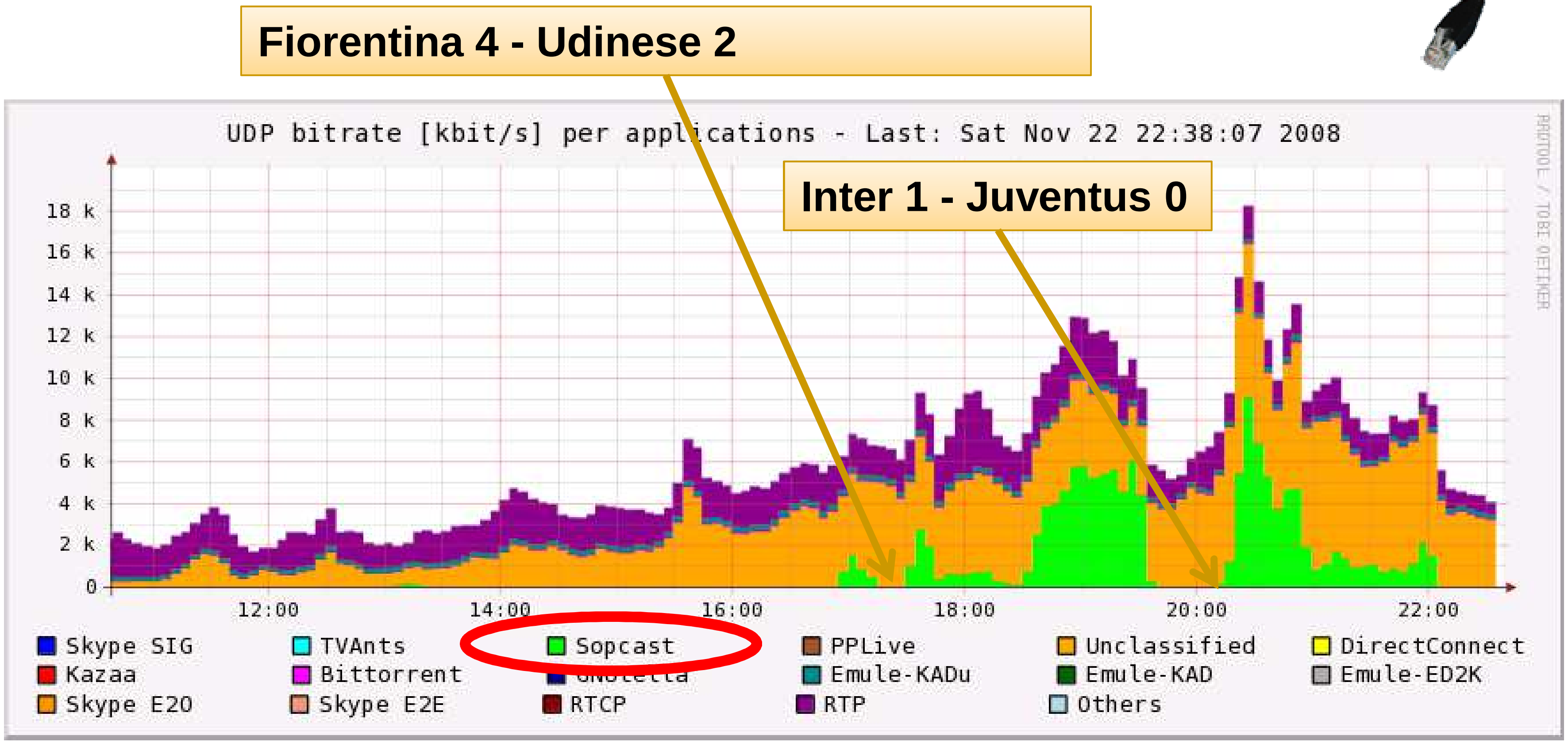


Anomalies (Good!)

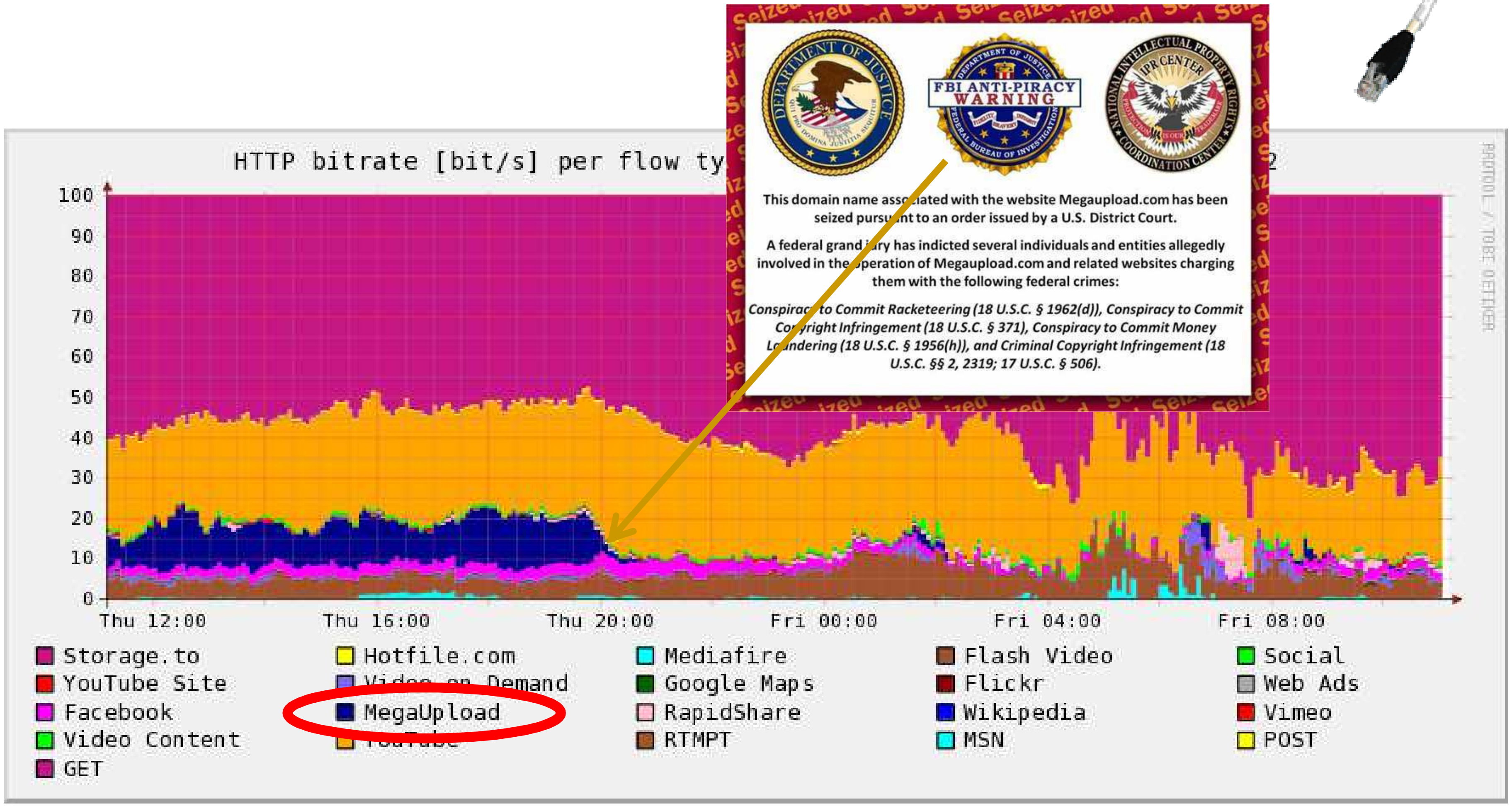
Spammer Disappear
McColo SpamNet shut off on
Tuesday, November 11th, 2008



New Applications – P2PTV



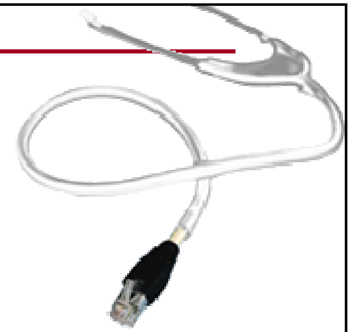
Megaupload blocked 19/01/12



How to monitor traffic?

- All previous examples rely on the availability of a CLASSIFIER
  - A tool that can discriminate classes of traffic
- Classification: the problem of assigning a class to an observation
  - The set of classes is pre-defined
  - The output may be correct or not

How to compute performance?



- Confusion matrix

		Predicted class		
		Cat	Dog	Rabbit
Actual class	Cat	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

  - On rows we have the actual class
  - On columns we have the predicted class
- Allows to see if some confusion arises

How to compute performance?

- Confusion matrix

		Predicted class		
		Cat	Dog	Rabbit
Actual class	Cat	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

- True positive
  - It was classified as a cat, and it was a cat

How to compute performance?

- Confusion matrix

		Predicted class		
		Cat	Dog	Rabbit
Actual class	Cat	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

- False negative
  - It was classified NOT as a cat, but it was a cat

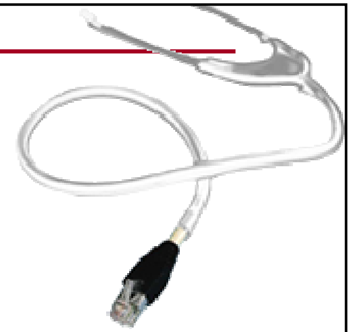
How to compute performance?

- Confusion matrix

		Predicted class		
		Cat	Dog	Rabbit
Actual class	Cat	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

- True negative
  - It was classified NOT as a cat, and it was NOT a cat

How to compute performance?

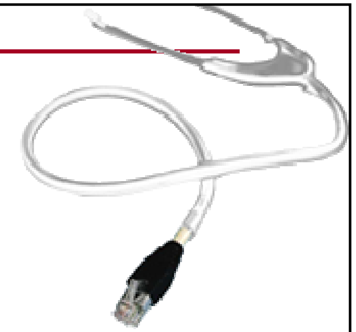


- Confusion matrix

		Predicted class		
		Cat	Dog	Rabbit
Actual class	Cat	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

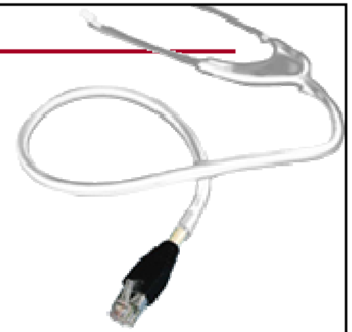
- False positive
  - It was classified as a cat, but it was NOT a cat

Other metrics



- Accuracy: is the ratio of the sum of all True Positives to the sum of all tests, for all classes.
- It is biased toward the most predominant class in a data set.
  - Consider for example a test to identify patients that suffer from a disease that affects 10 patient over 100 tests. The classifier that always returns ``sane'' will have accuracy of 90%.

Other metrics

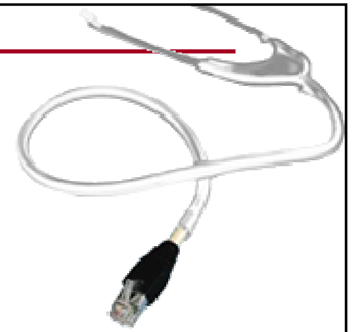


- Recall of a class: is the ratio of the True Positives and the sum of True Positives and False Negatives.
  - Recall(cat)=5/(5+3+0)

		Predicted class		
		Cat	Dog	Rabbit
Actual class	Cat	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

  - It is a measure of the ability of a classifier to select instances of the given class from a data set

Other metrics



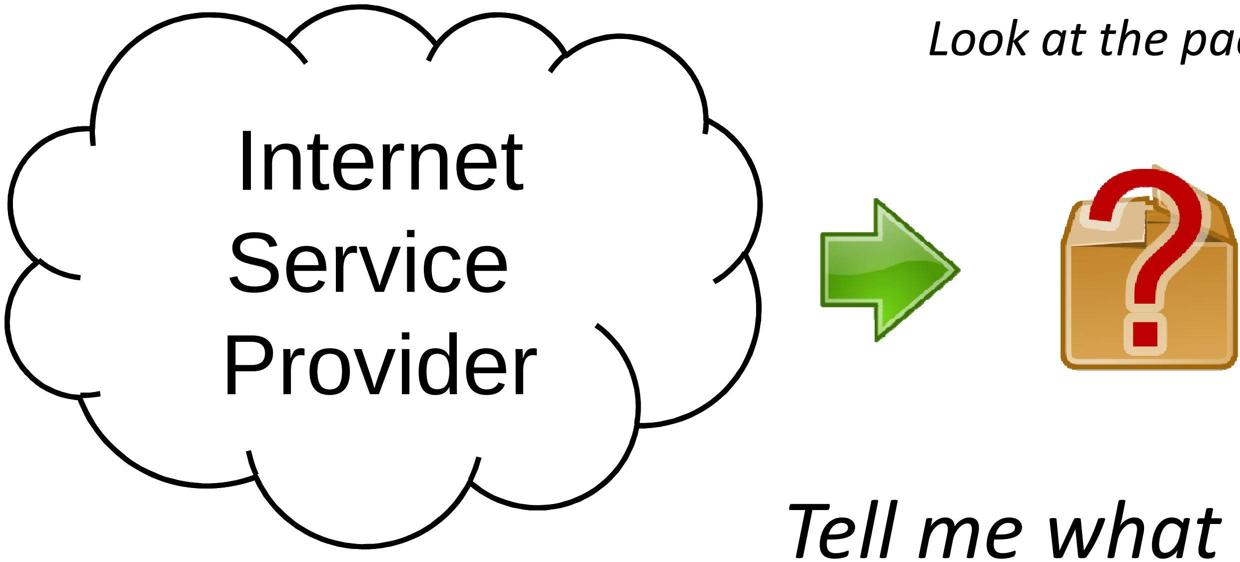
- Precision of a class: is the ratio of True Positives and the sum of True Positives and False Positive
  - Precision(cat) = 5/(5+2+0)

		Predicted class		
		Cat	Dog	Rabbit
Actual class	Cat	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

  - It is a metric that measure how precise is the classifier in labeling only samples of a given class

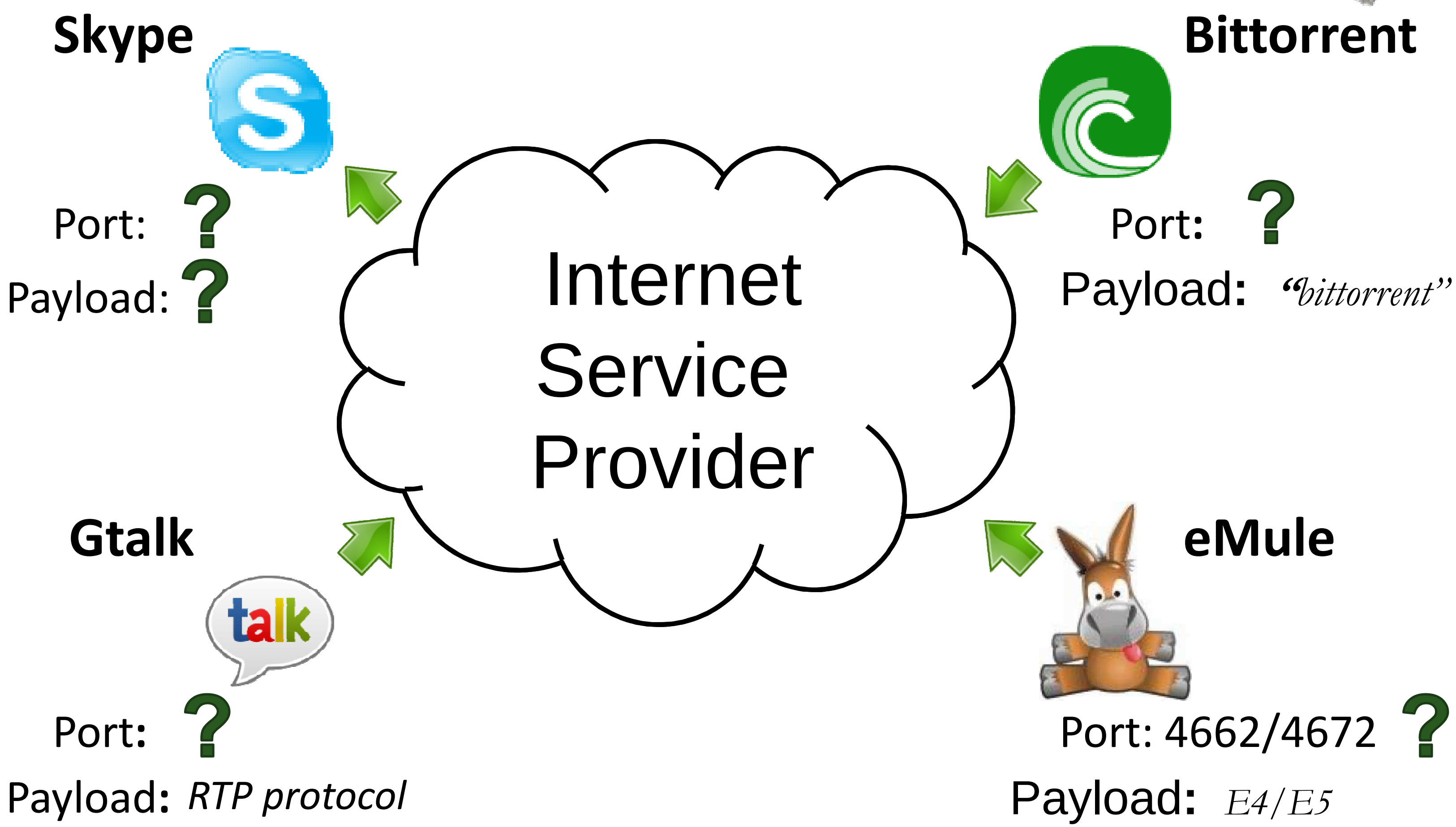
Traffic classification

Look at the packets...



Tell me what ***protocol*** *and/or* ***application*** *generated them*

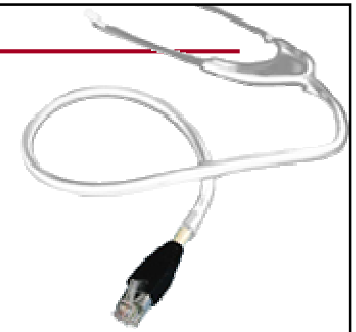
Typical approach:
Deep Packet Inspection (DPI)



The problem of traffic classification

- Deep Packet Inspection
  - Based on looking for some pre-defined payload patterns, deep in the packet
- Simple at L2-L4
  - “if ethertype == 0x0800, then there is an IP packet”
  - Usually done with a set of *if-then-else* or even *switch-case*
- Ambiguous at L7
  - TCP port 80 does not mean automatically “protocol HTTP”

DPI: Rule-set complexity



- Practical rule-sets:
  - Snort, as of November 2007
    - 8536 rules, 5549 Perl Compatible Regular Expressions
  - OpenDPI as of February 2012 (more protocols added recently → paper)
    - 118 protocols
  - Tstat as of February 2012
    - Approx 200 classes/services

Deep packet inspection = Regular expression matching at line rate ⇨ Finite Automata based techniques

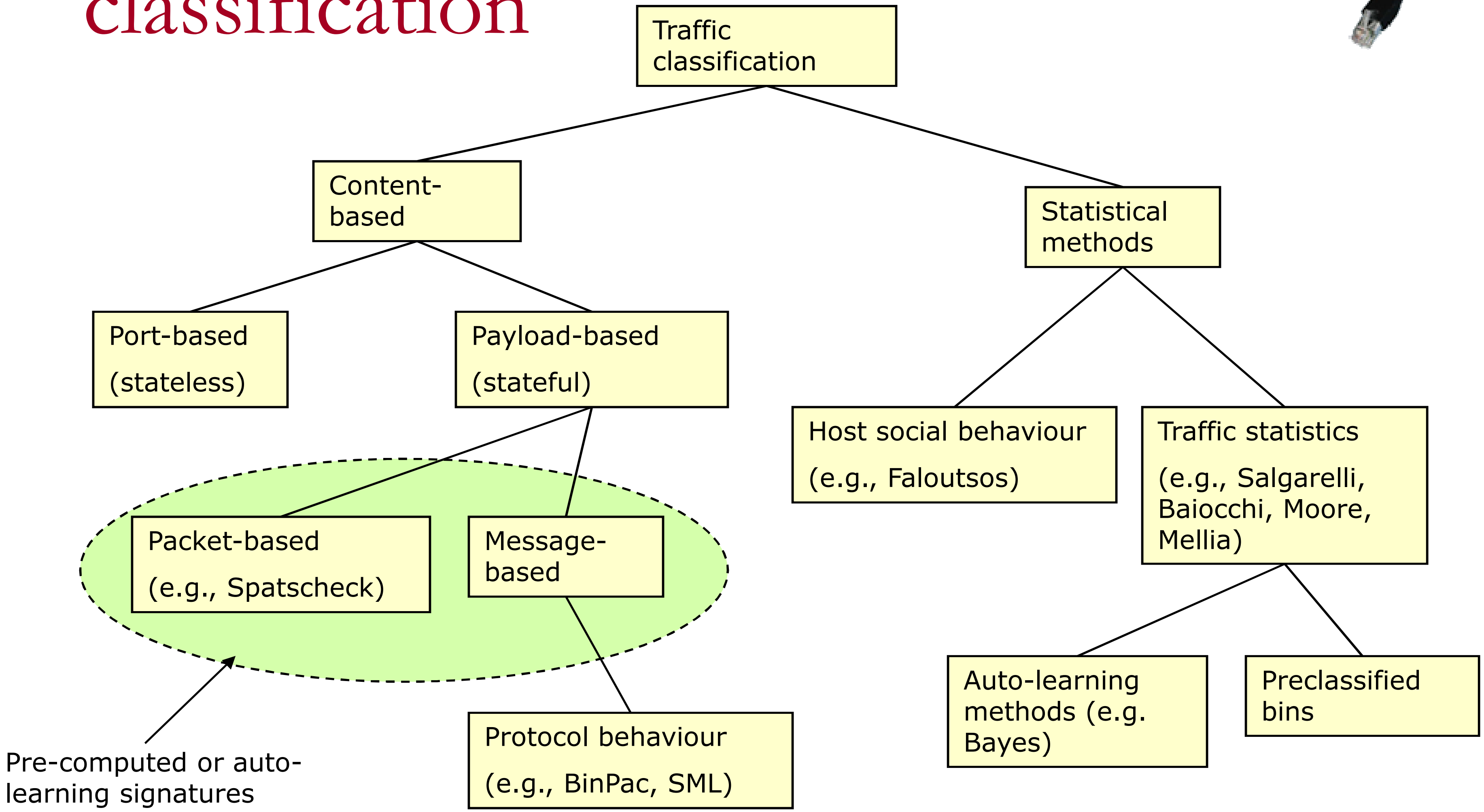
Some notes...

- Protocol identification…
- … or application verification?
  - Skype can use the standard HTTP protocol to exchange data
  - Is that traffic “Skype” or “HTTP”?
- Today everything is going over HTTP
  - Is it Facebook? Twitter? YouTube video? Or HTTP?

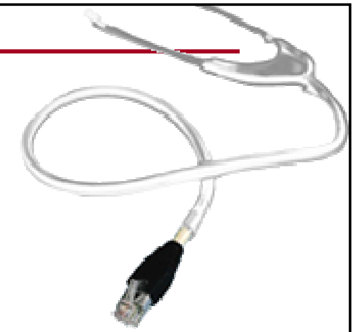
The question

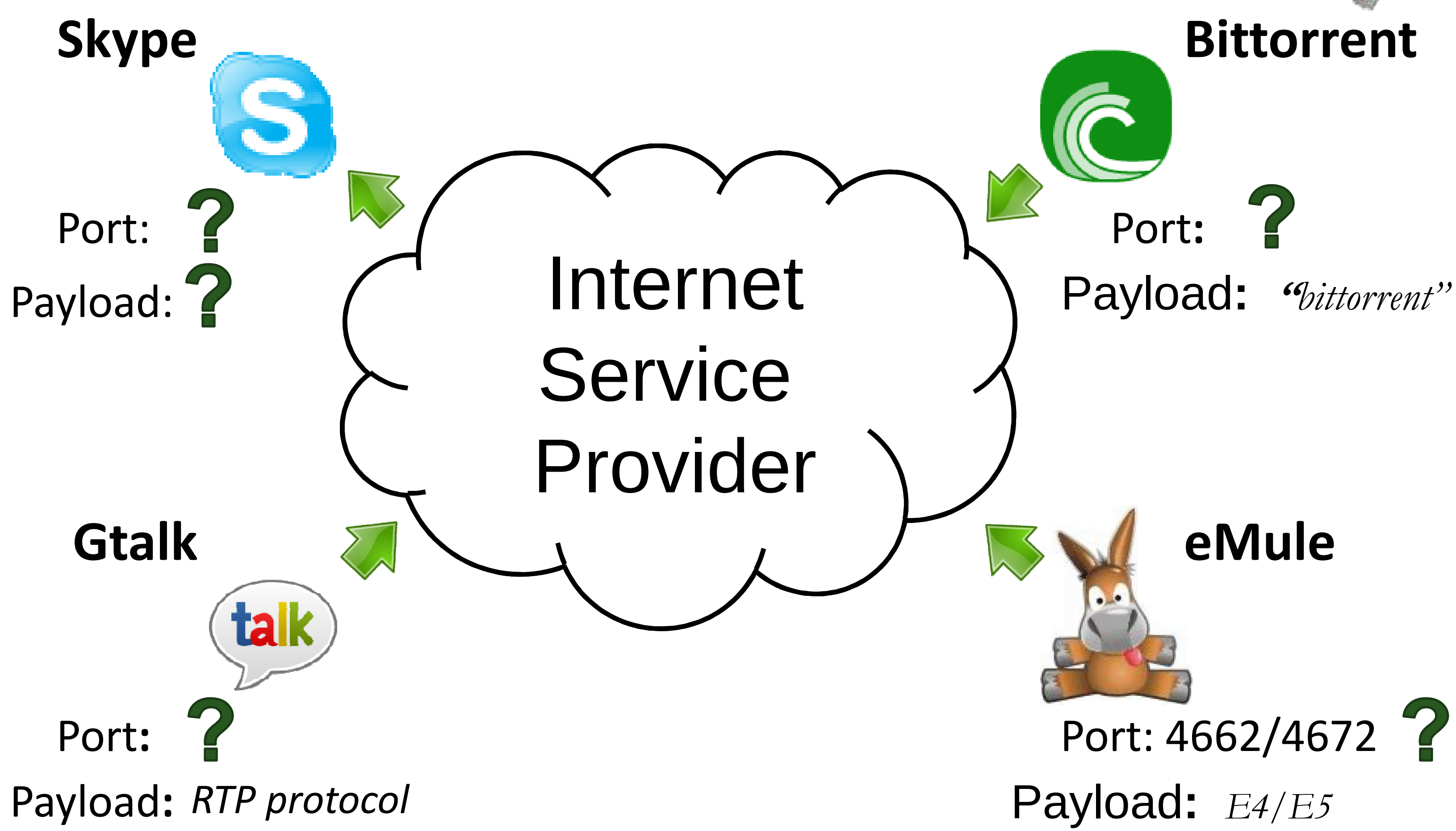
Which granularity are you interested into ??

Several approaches to traffic classification



Typical approach:
Deep Packet Inspection (DPI)





Typical approach: Deep Packet Inspection (D…

Bittorrent

Port: ?

…rrent"

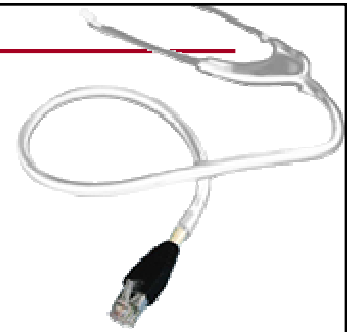
It fails more and more:
P2P
Encryption
Proprietary solution
Many different flavours

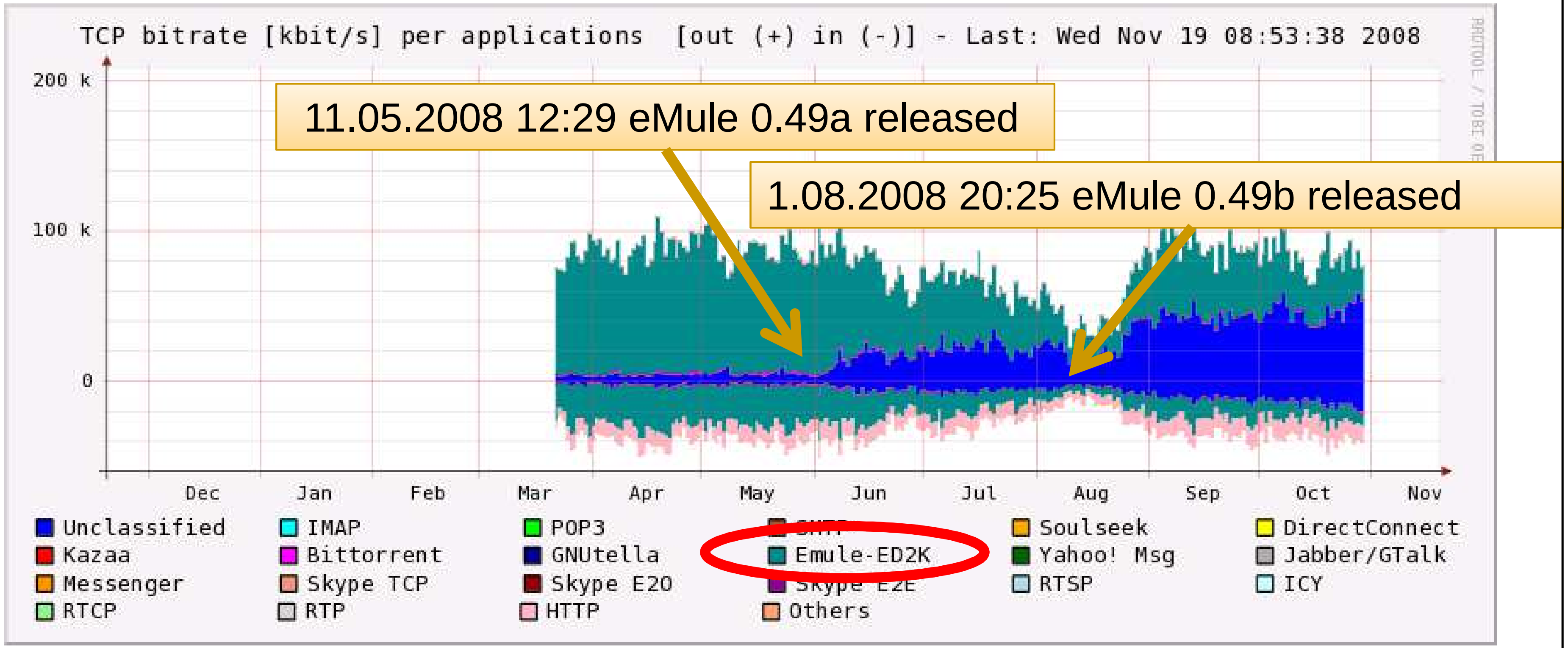
eMule

Port: ?
Payload: *RTP protocol*

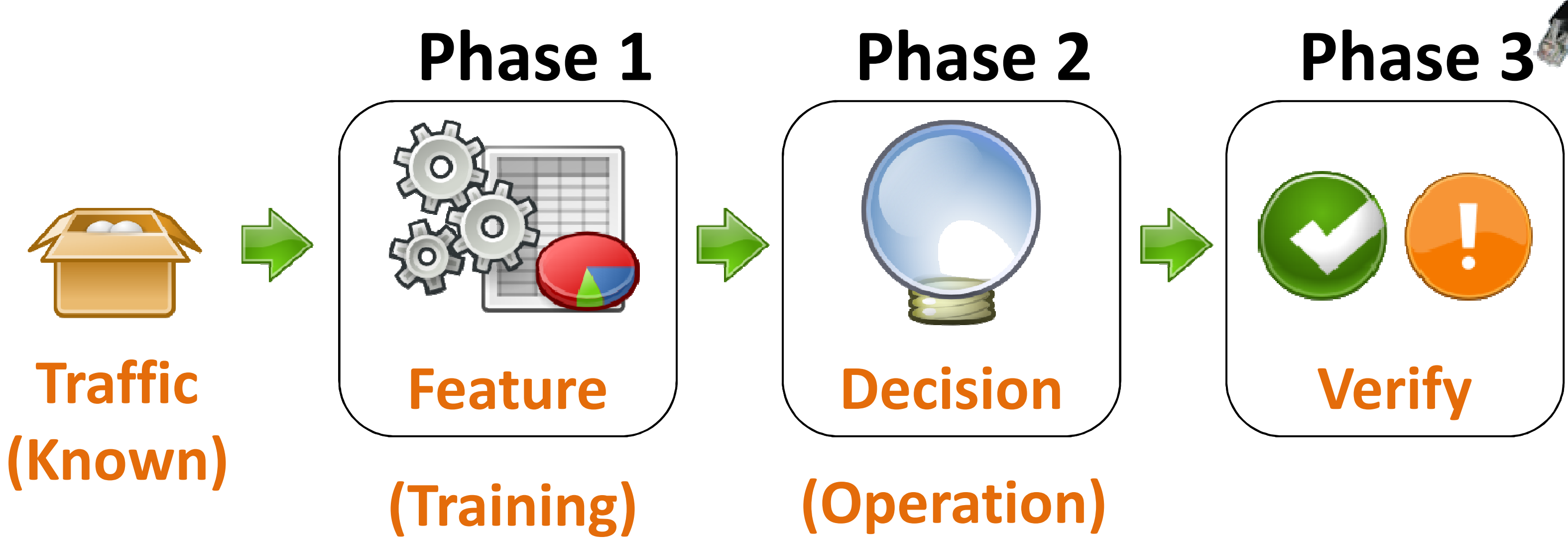
Port: …2/4672 ?
Payload: *E4/E5*

The Failure of DPI





Possible Solution: Behavioral Classifier



1. **Statistical** characterization of traffic
2. Look for the **behaviour** of unknown traffic and assign the class that better fits it
3. Check for possible classification mistakes

Behavioural classifiers

- Which statistics?
  - Packet size
    - Average, std, max, min
    - Len of first X pkts
  - IPG
    - Average, std, max, min
    - IPG of first X+1 pkts
  - Total size, duration, #data packets
  - From client, from server, from both
  - RTT, #concurrent connection, rtx, dups, …
  - TCP options, flags, signaling, …
- Feature selection?
- Which decision process?
  - Ad Hoc
  - Bayesian
  - Neural Networks
  - Decision trees
  - SVM
  - …
- Which training set?
  - Supervised techniques

The case of Skype

Our Goal

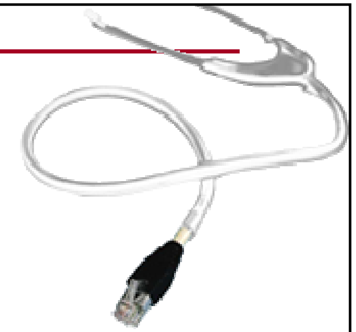
- Identify skype traffic
- Motivations
  - Operators need to know what is running in their network
    - New business models, provisioning, TE, etc.
  - Understand user behaviour
  - Traffic characterization, security
  - …
  - It's fun

Skype Overview

State-of-the-Art Encryption/Obfuscation Mechanisms

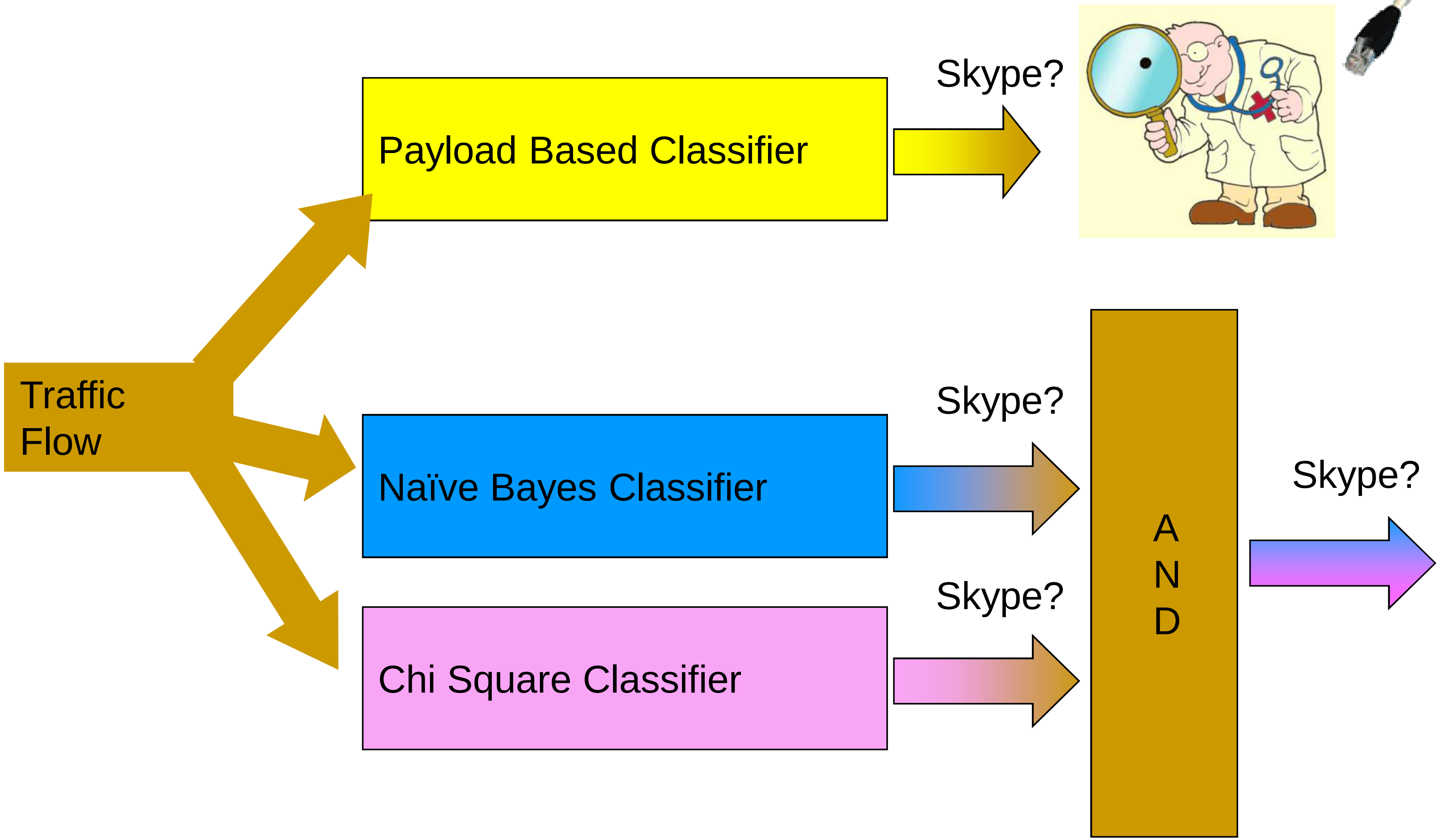
- Skype offers vo
  transfer service
- Closed design, proprietary solutions
  - P2P technology
  - Proprietary protocols
  - Encrypted communications
- Easy to use, difficult to reveal
  - It is the perfect example of DPI failure

Our Goal



- Identify Skype traffic
  - Voice stream first: both E2E and SkypeOut/In streams
  - Possible video/chat/file transfers/signaling
- Constraints
  - Passive observation of traffic
  - Protocol ignorance

Three Classifiers

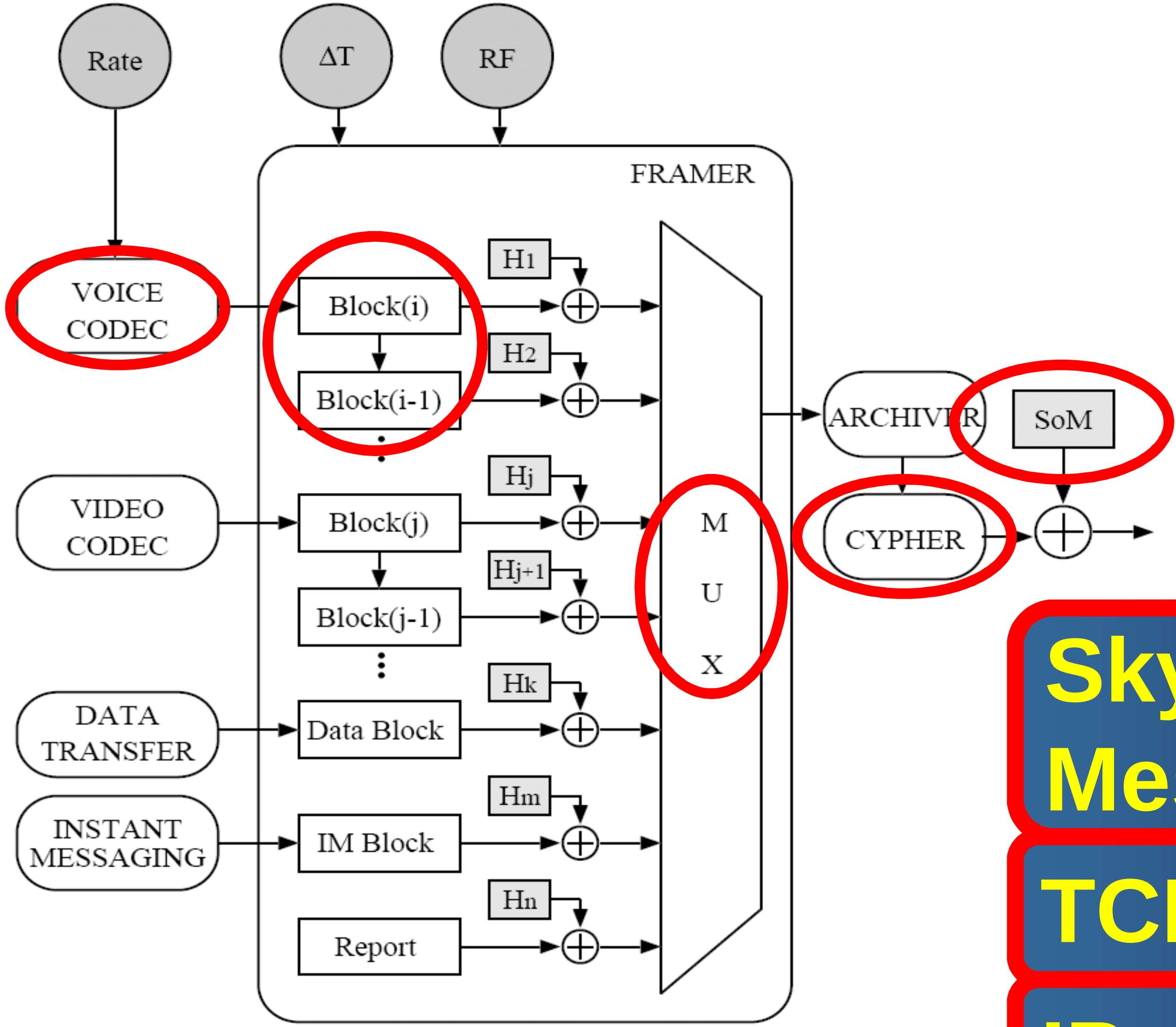


Phase 1 – try to understand it

Skype as VoIP Application

- Skype selects the voice codec from a list
  - Low bit rate: 10-32 kbps
  - Regular Inter-Packet-Gap (30 ms frames)
- Redundancy may be added to mitigate packet loss
- Framing may be modified from the original codec one
- Multiplexes different source into the same message (voice, video, chat,…)

Skype Source Model



Skype Header Formats (What we guess about it)

Can we design a DPI classifier?

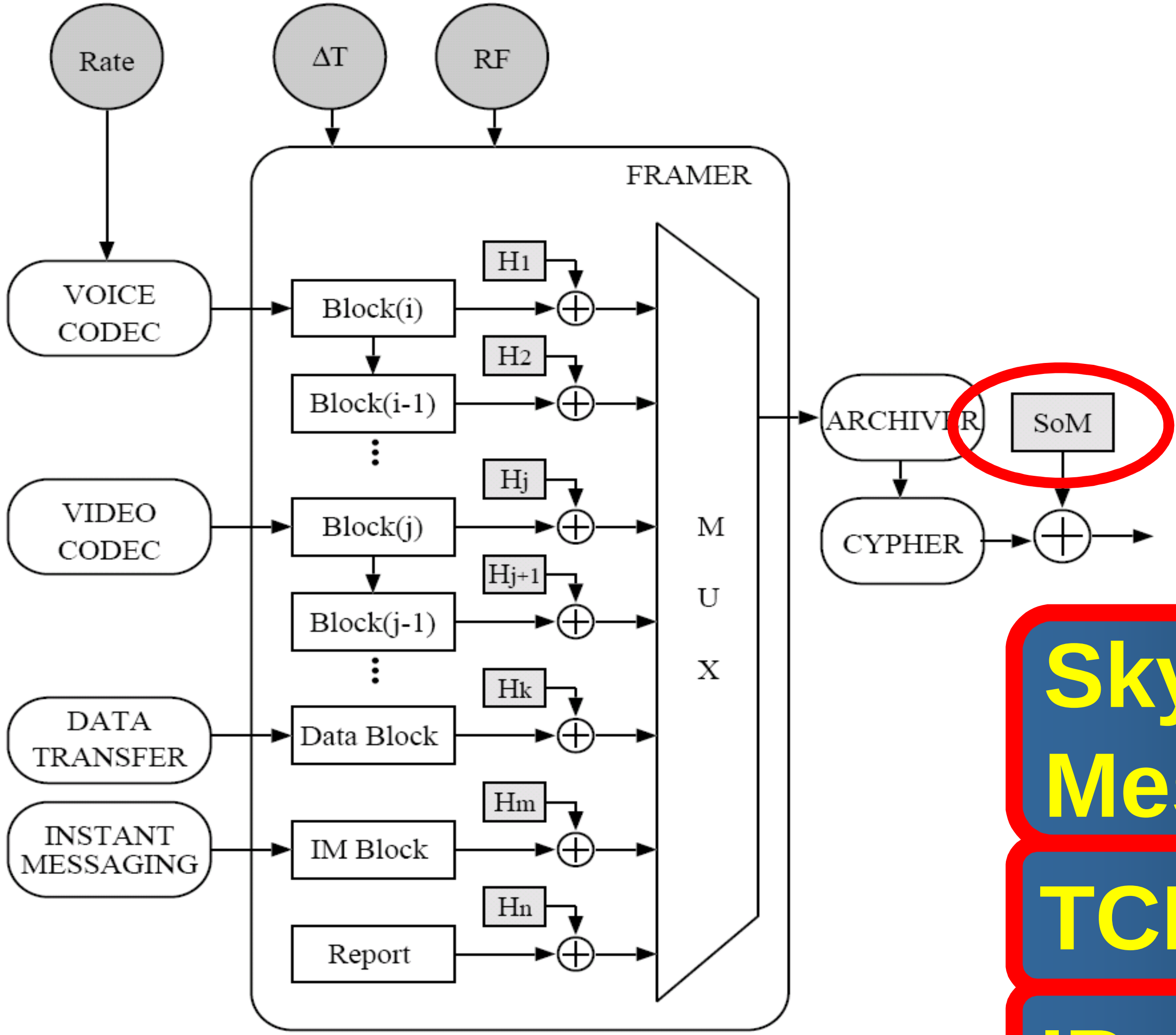
Possible Skype Messages

- Signaling and data messages
  - Use TCP, with ciphered payload
    - Login, lookup, signaling…
    - Data flow
  - Use UDP whenever possible: payload is encrypted … but…

Impossible to exploit. Everything is ciphered

- **Some header MUST be exposed…**

Skype Source Model



Skype Message

TCP/UDP

IP

SoM Format for E2E Messages

```
0       8       16      24
---------------------------
|       ID      |  FUNC   |
---------------------------
```

Start of Message (SoM) of End2End messages carried by UDP has:

- `ID`: 16 bits long random identifier
- FUNC: 5 bits long function (multiplexing?), obfuscated in a Byte

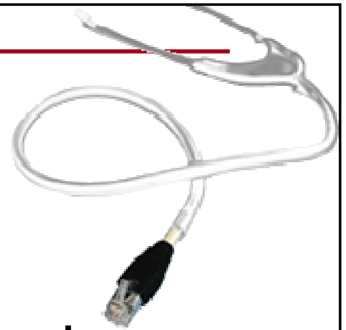
Function Values

- 0x01 = ??Query message
- 0x02 = ??Query
- 0x0d = Data
- 0x07 = NAK

Voice
Video
Chat
File

```
No.        Time         Source
      343 0.055176      83.225.12.108

0000   0a a0 4d 4c 98 f5 28 bb 8a 90 55
0010   e2 da d7 7b 4f 84 b3 6e a7 1e 4c

      344 0.060192      83.225.12.108

0000   8c d9 4d fa 67 cc 83 fc 7f a2 1b
0010   d6 72 41 36 62 d1 b8 72 50 25 6c

      345 0.061195      83.225.12.108

       3d 25 6d 8e 12 37 f9 84 b8 f1 e4
       15 c7 68 37 25 01 3a 6f 11 aa 92

      346 0.064205      83.225.12.108

       10 85 0d 44 a3 f2 45 91 b8 3c 5d
       64 e8 07 aa 59 da 24 76 fa bb 5a

      347 0.058186      83.225.12.108

0000   b2 51 0d 5c db 86 59 d4 53 59 fe
0010   39 d3 12 12 84 9e 94 78 26 12 90
      359 0.059188      83.225.12.108
```

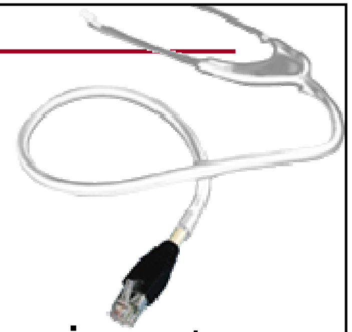
PBC



- SoM can be used to identify Skype flows carried by UDP
  - 5bits long signature

Classic signature based classifier

PBC



- SoM can be used to identify Skype flows carried by UDP
  - 5bits long signature

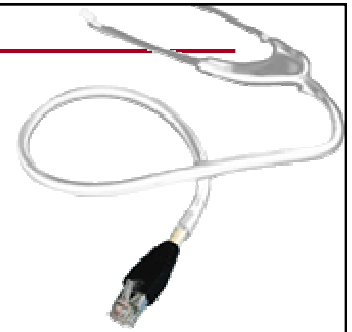
Classic signature based classifier

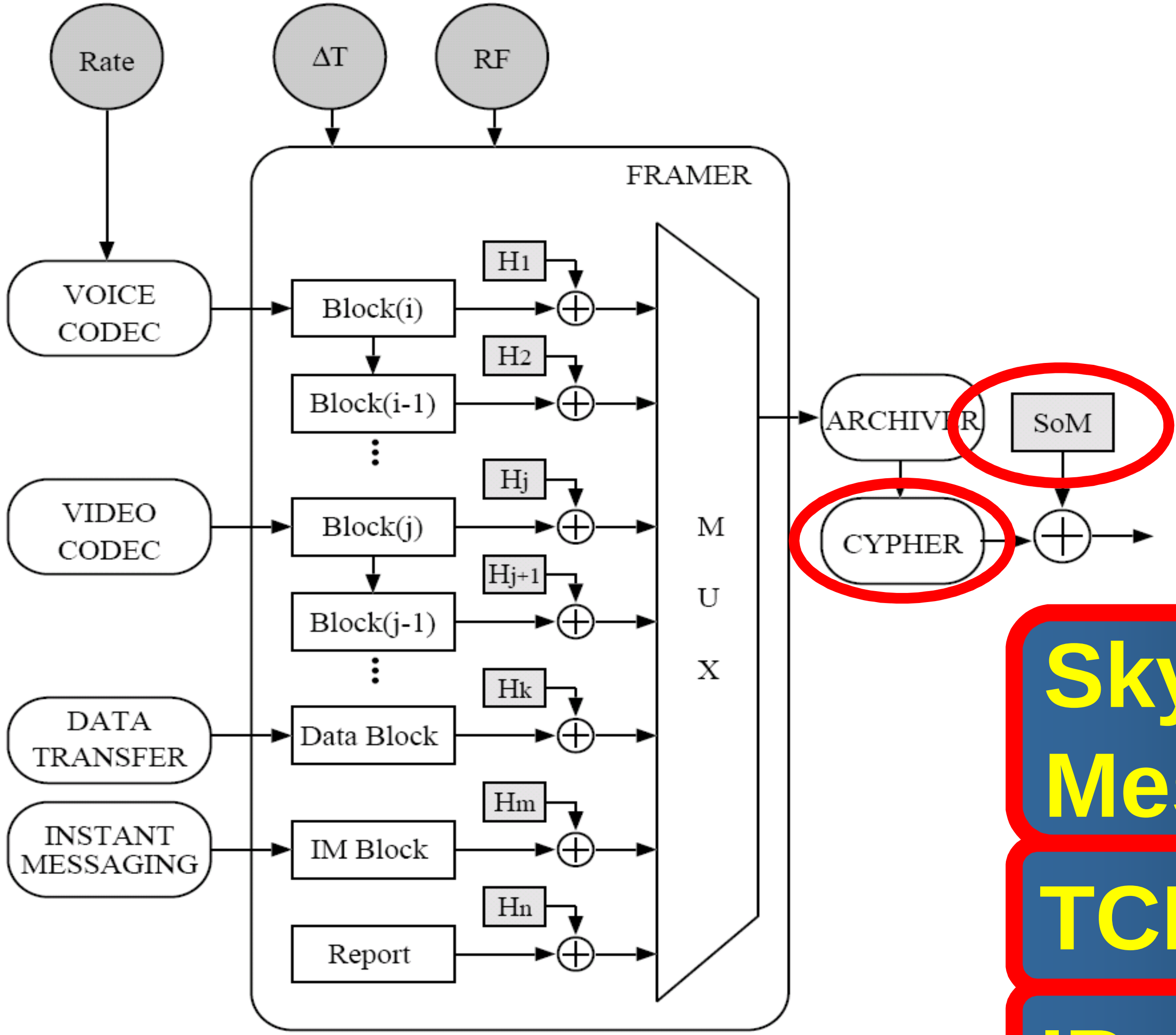
- IMPROVE: Identify Skype socket address at clients
  - The UDP port is FIXED and not random (as in TCP)
  - Then, look for Skype flows with the same UDP port
- It works
  - with UDP only
  - at edge node only
- Cannot discriminate VOICE/VIDEO/CHAT/DATA

Skype Encrypts Traffic

Can we leverage this?

Skype Source Model



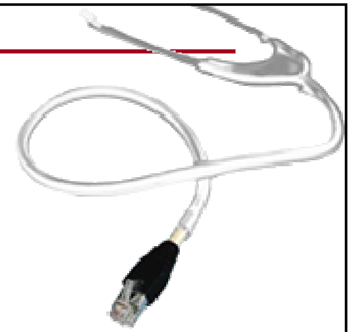


Skype Message

TCP/UDP

IP

Randomness Classifier

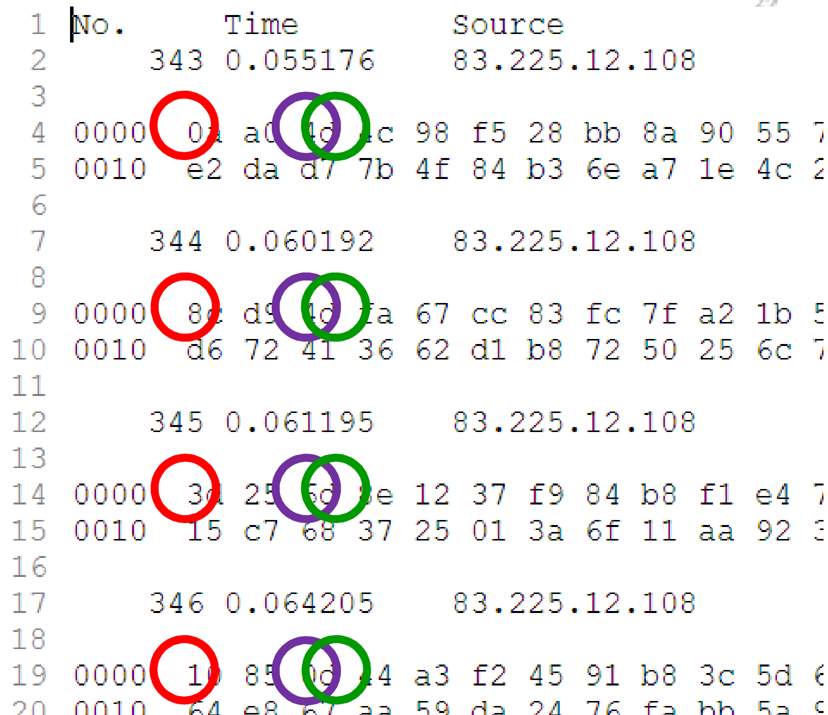


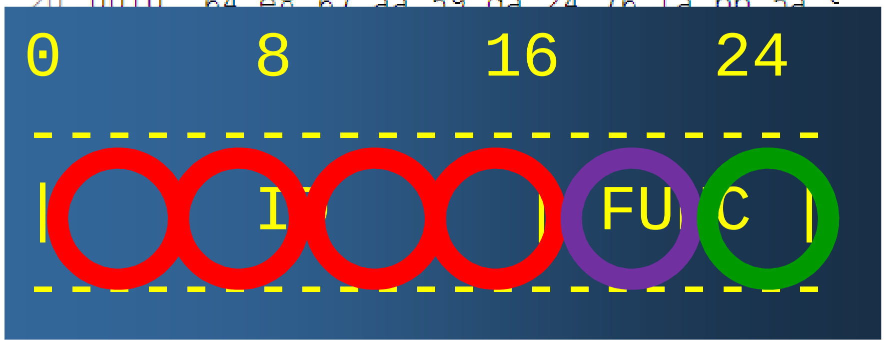
- Skype encrypts traffic
  - payload looks like random
- Some headers are constant (FUNC)
- Apply randomness test to the payload bits
  - Chi-Square test:
    statistic test for random sequences

$$\chi^2 = \sum_i \frac{(x_i - E)^2}{E}$$

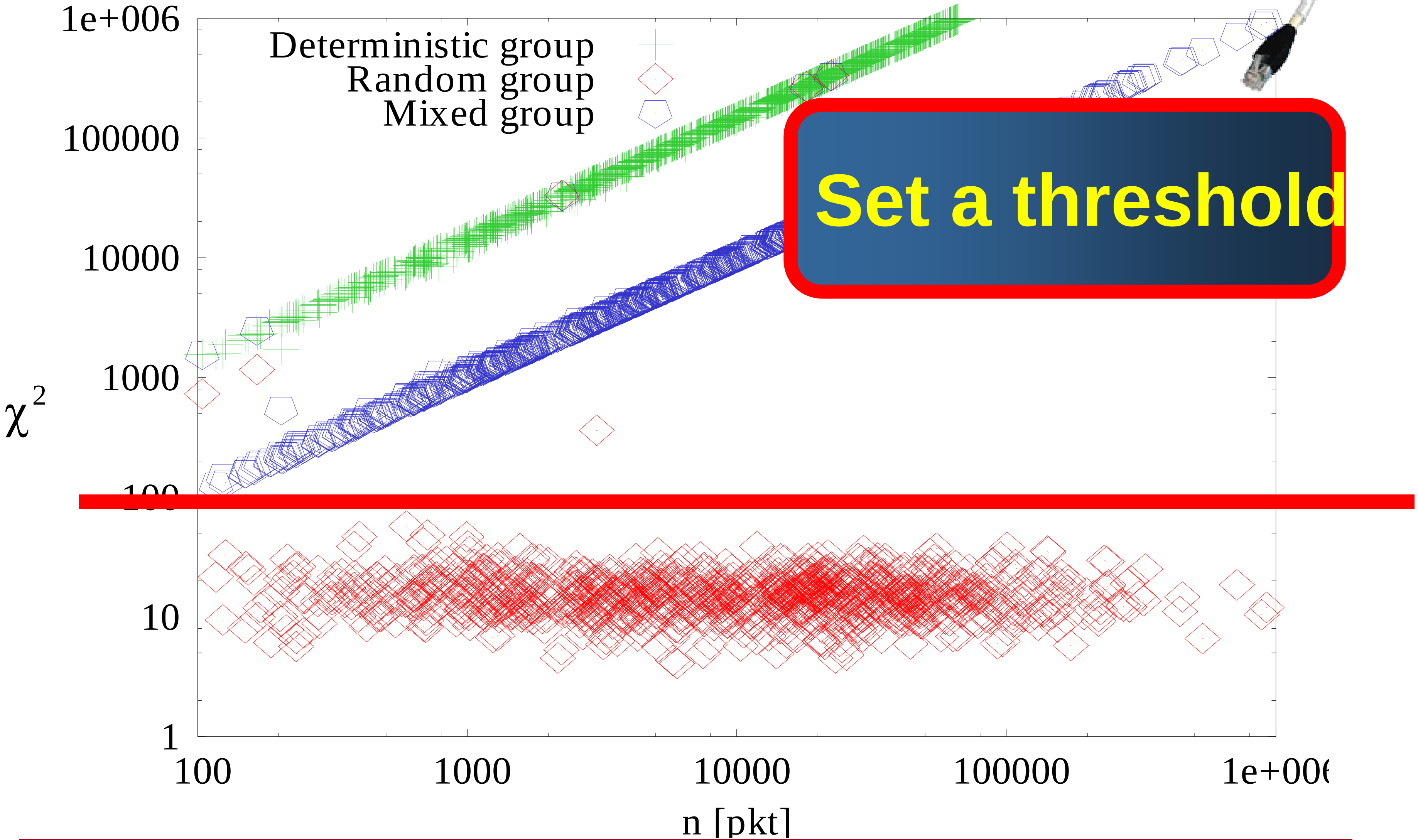
Randomness Classifier

- Split the payload into groups
- Apply the test on the values assumed at each group
  - Each message is an observation
- Some groups will contain
  - Random bits
  - Mixed bits
  - Deterministic bits





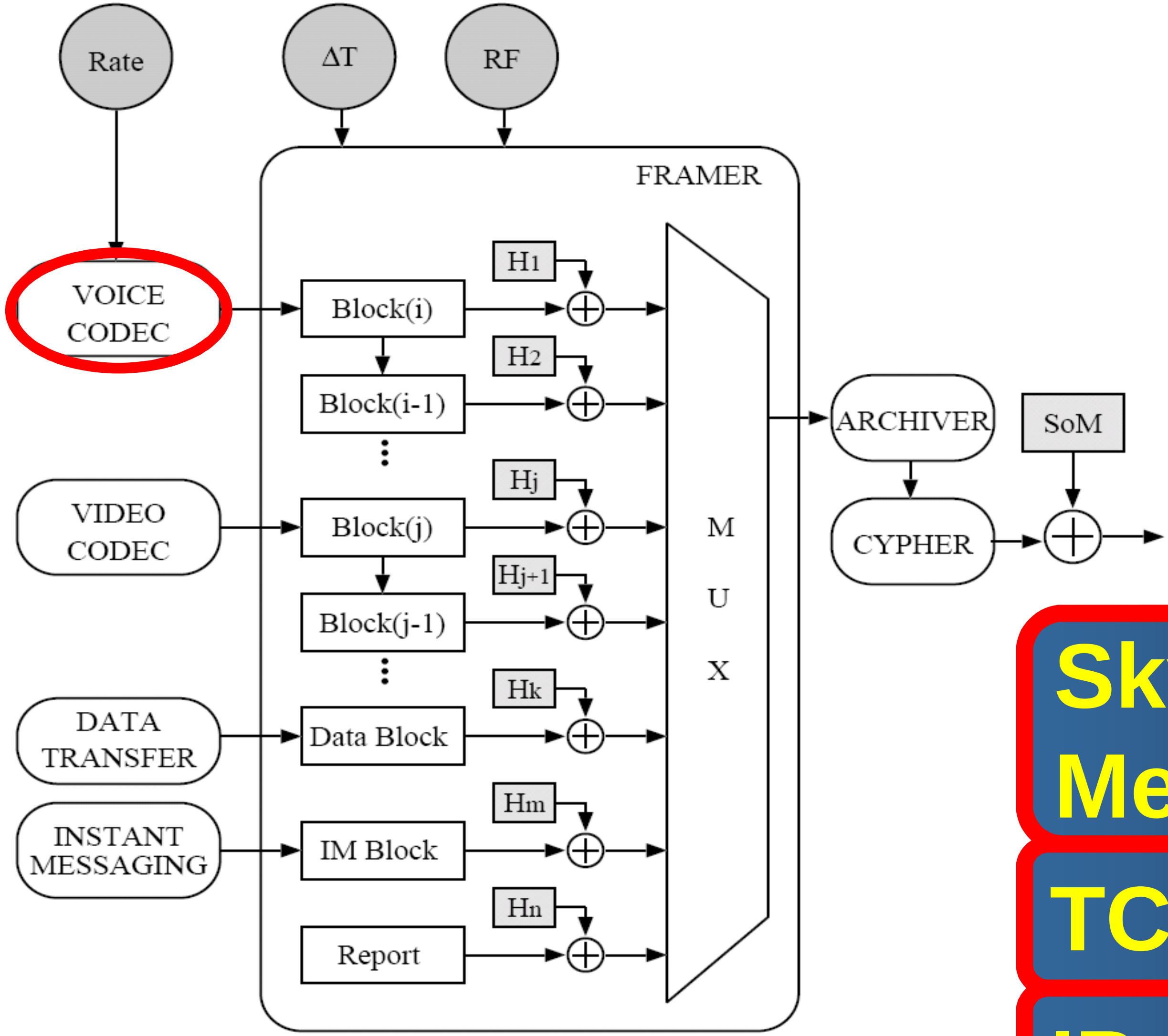
Randomness Classifier



Skype is a VoIP Application

Which are the features that make it different from a bulk download?

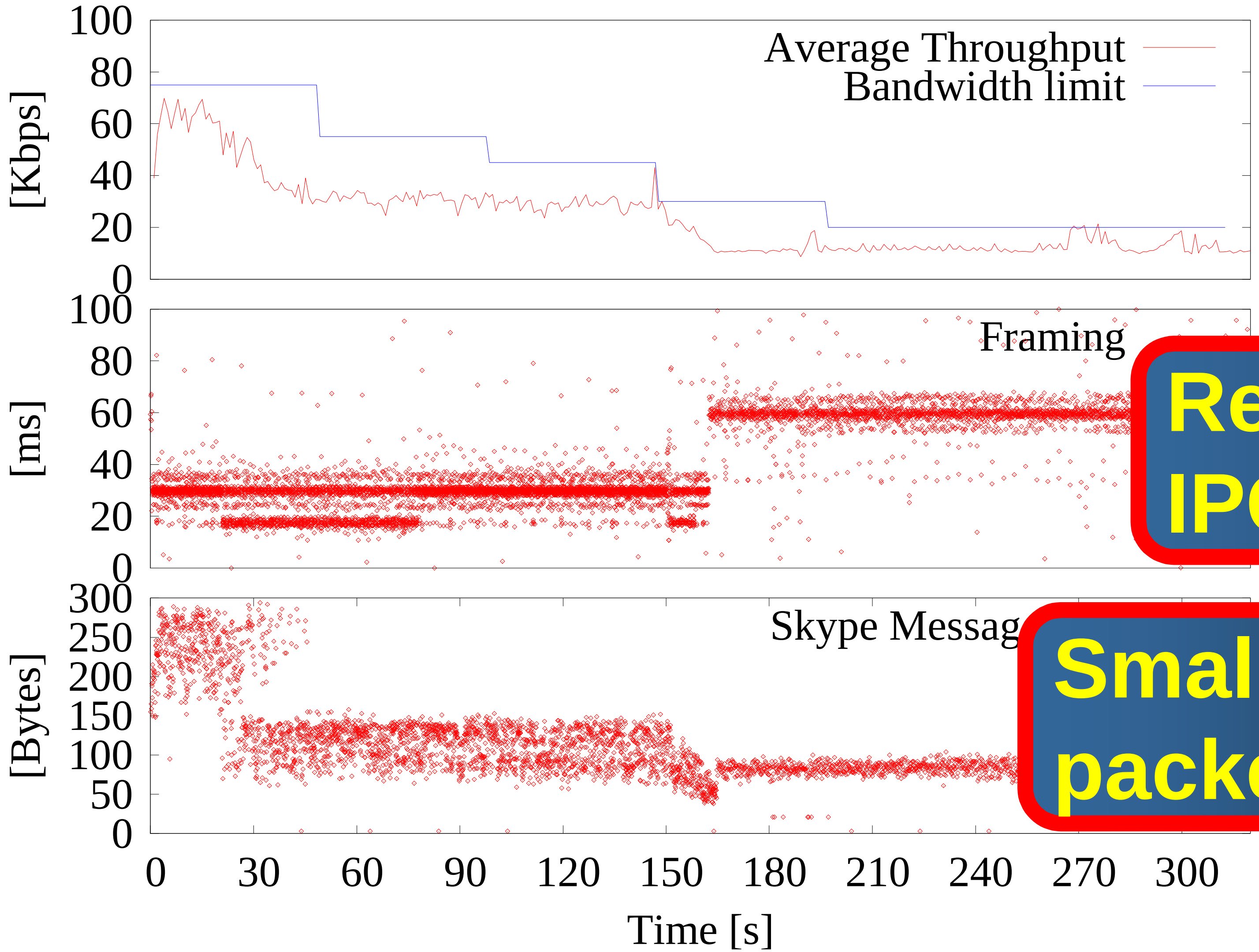
Skype Source Model



Which features?

- **Question: Which features would you select to differentiate a VoIP stream from a data download?**

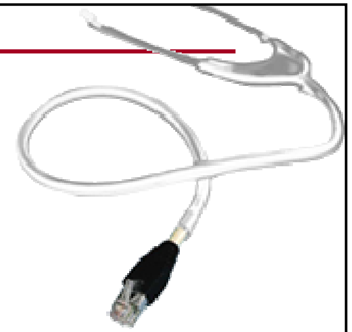
Sample Trace



Regular IPG

Small/regular packets

Naive Baysean Classifier

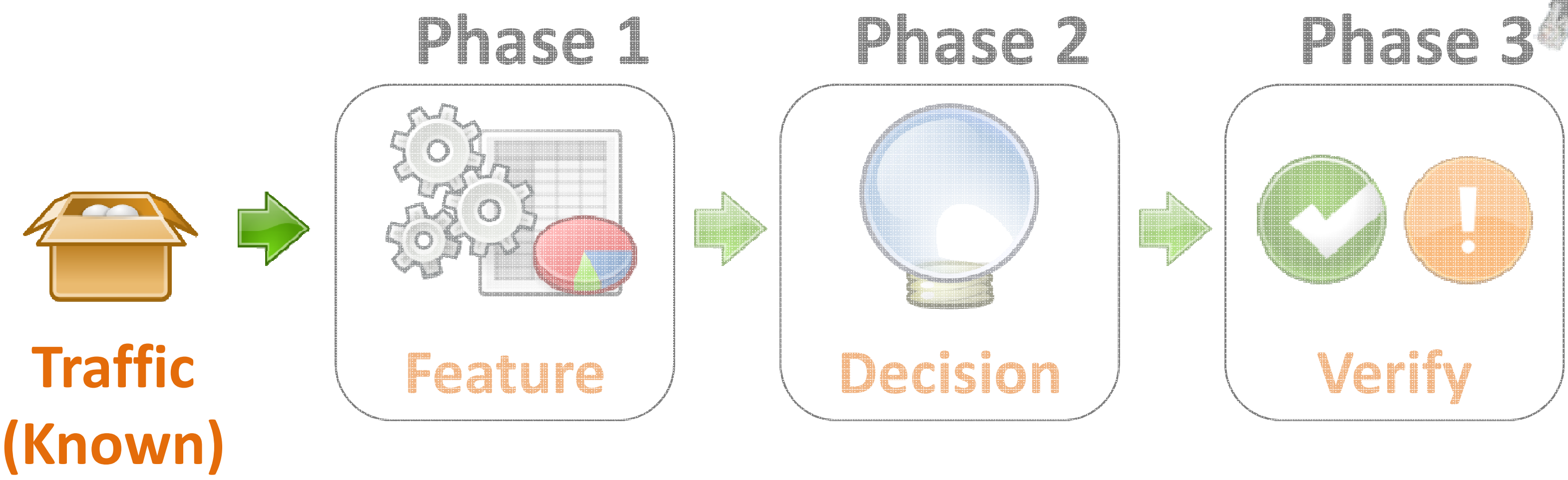


- Simple classifier: based on the a-priori prob, evaluate the a-posteriori prob
  - How similar is this flow to a Skype voice flow?
- What makes “VoIP” traffic different from other traffic?
  - Packet size, i.e., small packets (packet NBC)
  - Inter-Packet-Gap, i.e., small IPG (IPG NBC)

Kiss: chi square stocastich classifier or Stocastic Packet Inspection

Generalize it

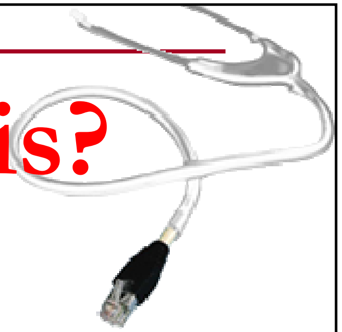
Statistical Approach



- Statistical characterization of bits in a flow
  - χ^2 Test

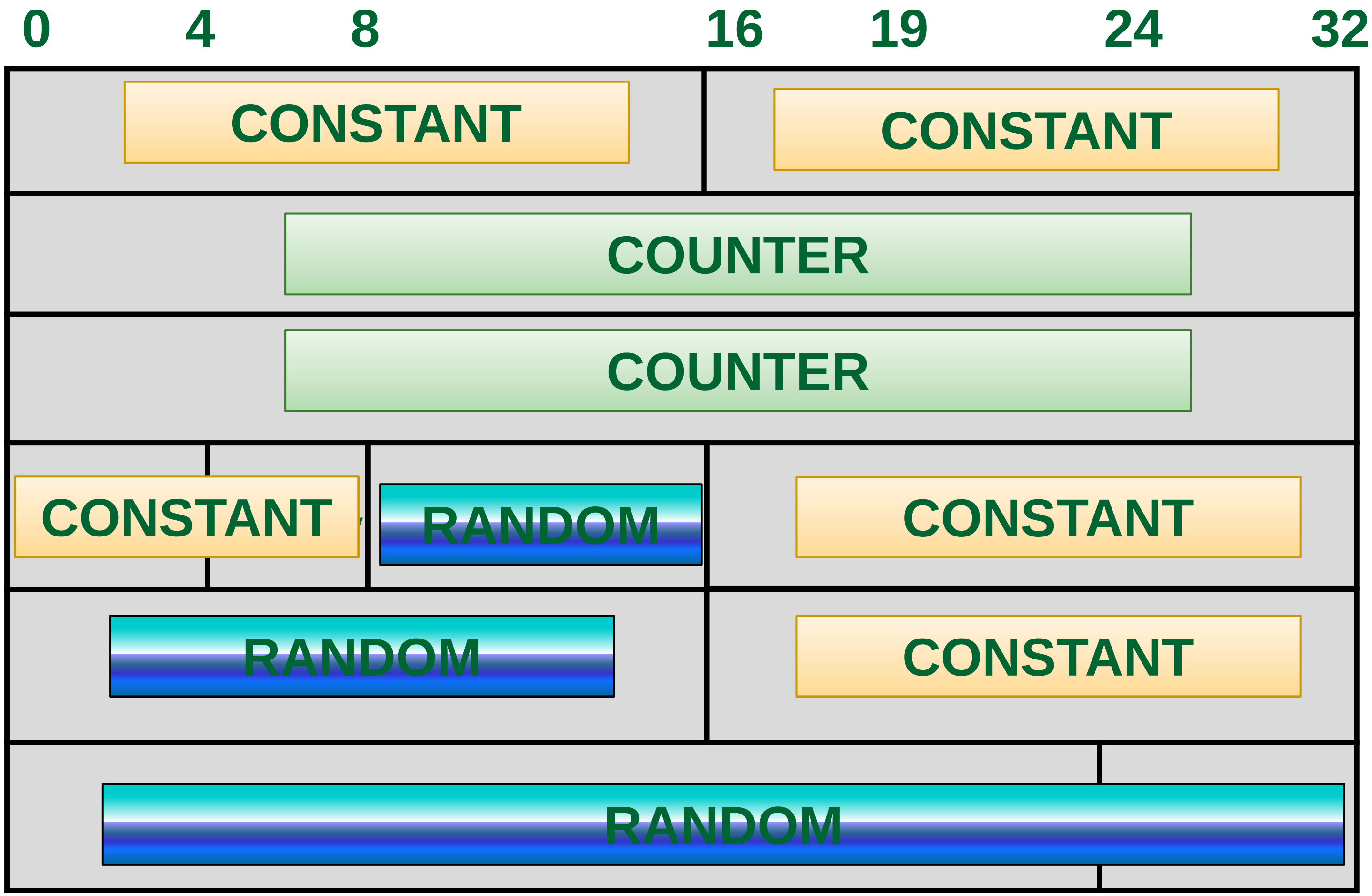
- Do ***NOT*** look at the ***SEMANTIC*** *and* ***TIMING***
- … but rather look at the protocol ***FORMAT***

Question: Which protocol is this?



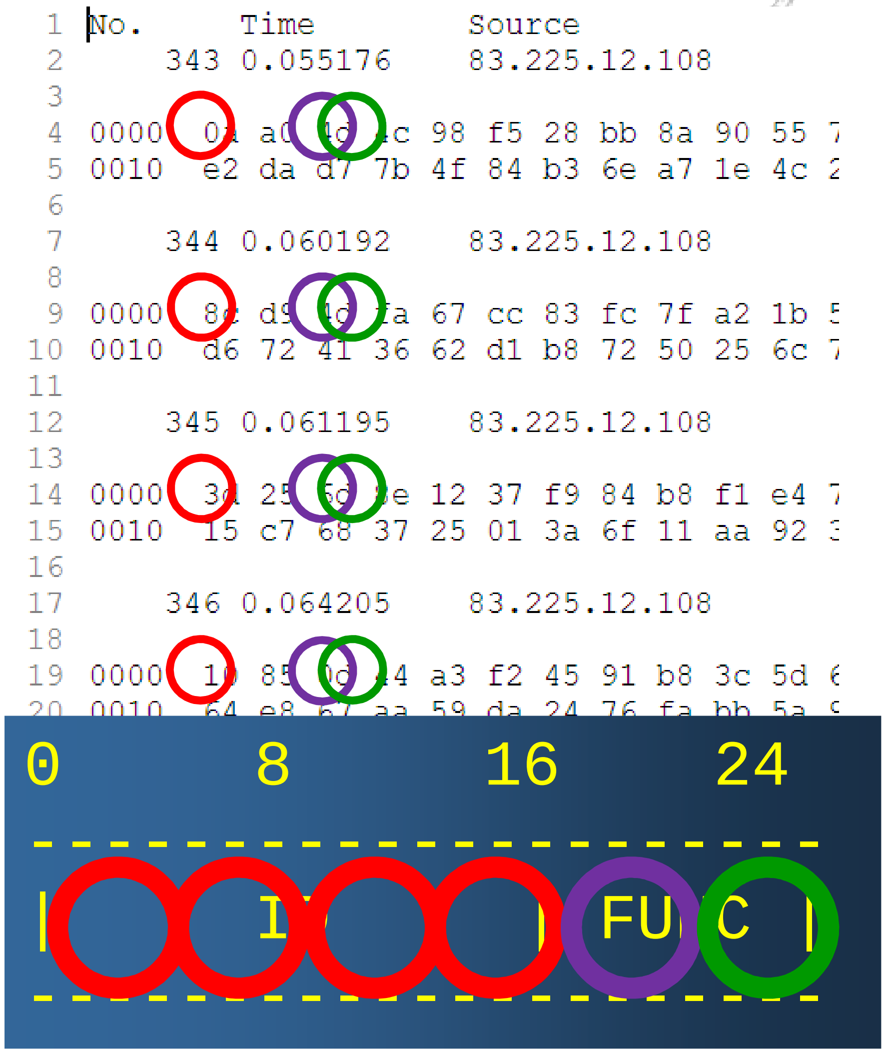
0	4	8	16	19	24	32
Source Port			Destination Port			
Sequence Number						
Acknowledgment Number						
HLEN	Resv	Control flag	Window			
Checksum			Urgent Pointer			
Options					Padding	

Question: Which protocol is this?

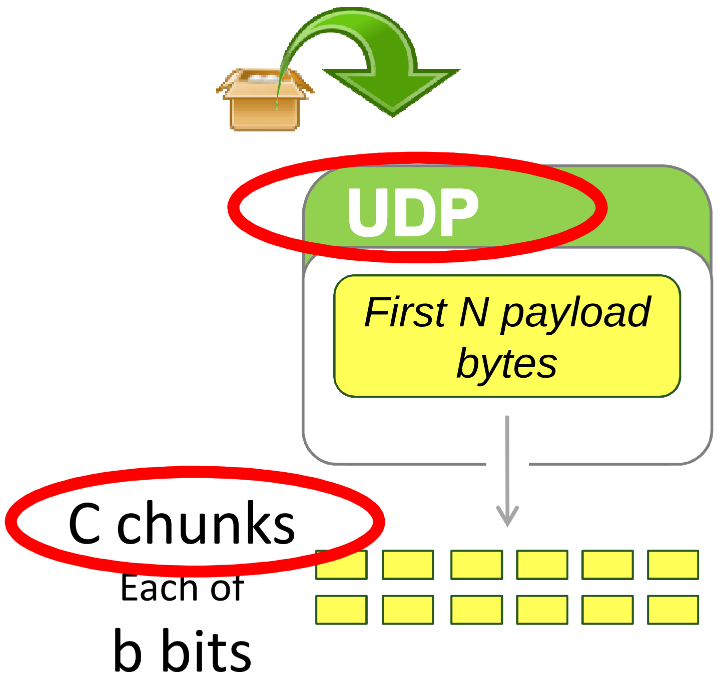


Randomness Classifier

- Split the payload into groups
- Apply the test on the values assumed at each group
  - Each message is an observation
- Some groups will contain
  - Random bits
  - Mixed bits
  - Deterministic bits



Chunking and χ^2



Observed distribution

Expected distribution (uniform)

$$\chi^2 = \sum_{i=1}^{2^b} \frac{(O_i - E_i)^2}{E_i}$$

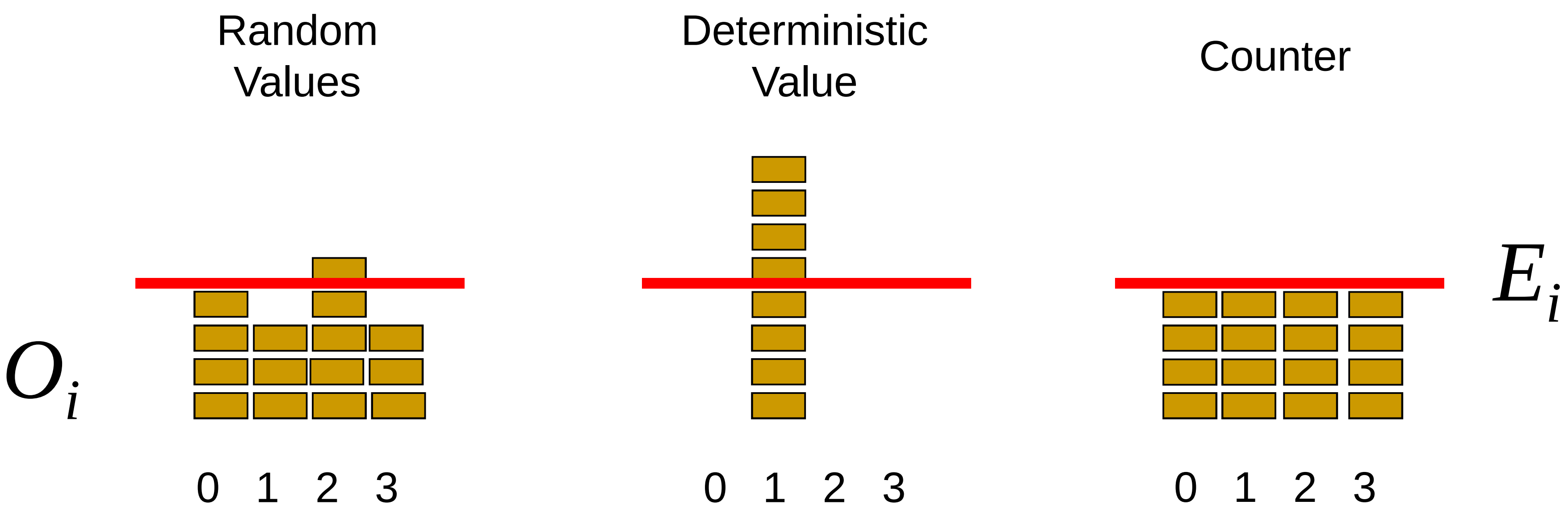
$$[\chi^2_1, \ldots, \chi^2_C]$$

Vector of Statistics

The χ^2 provides an implicit measure of *entropy* or *randomness*

Consider a chunk of 2 bits:

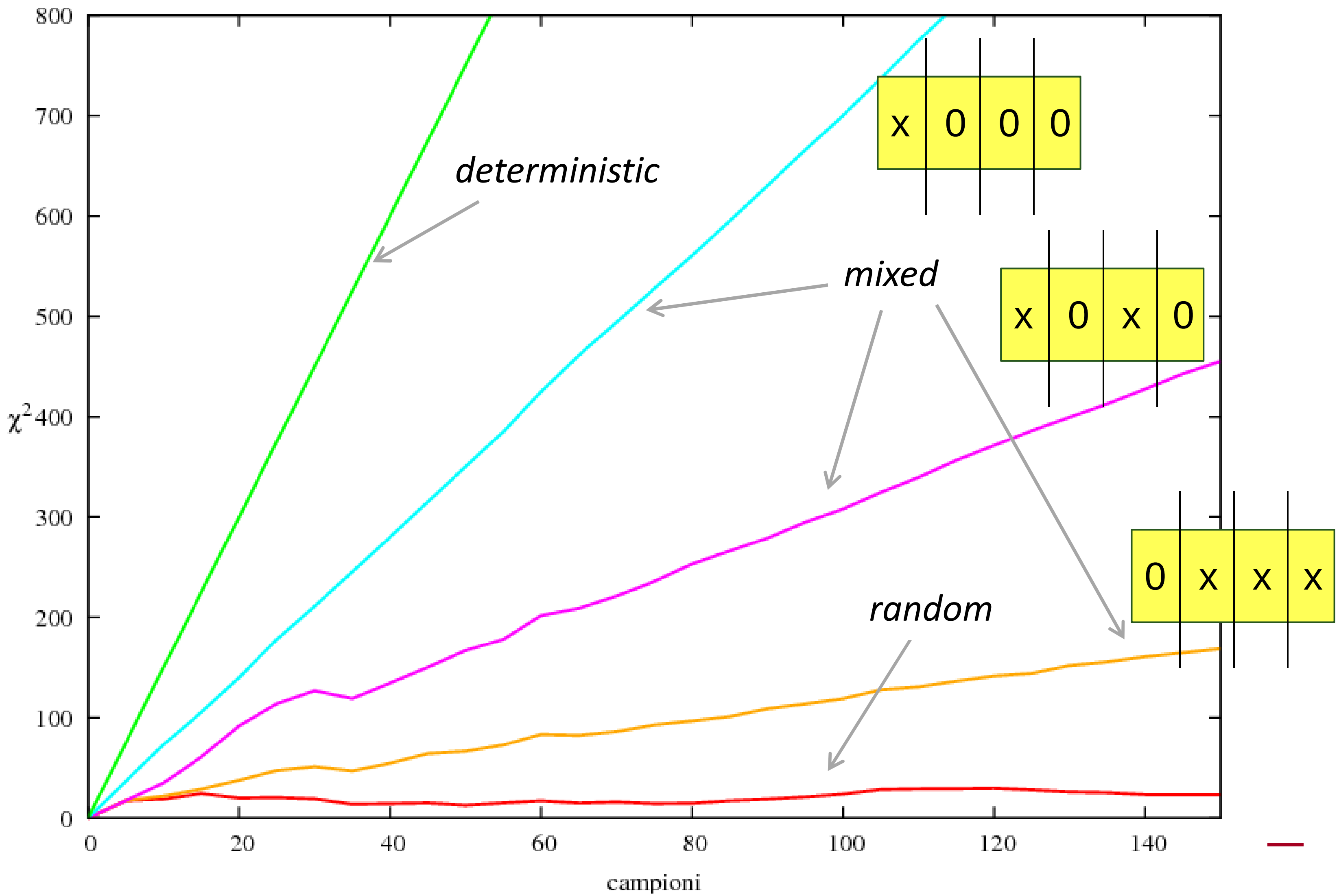
and different behaviour



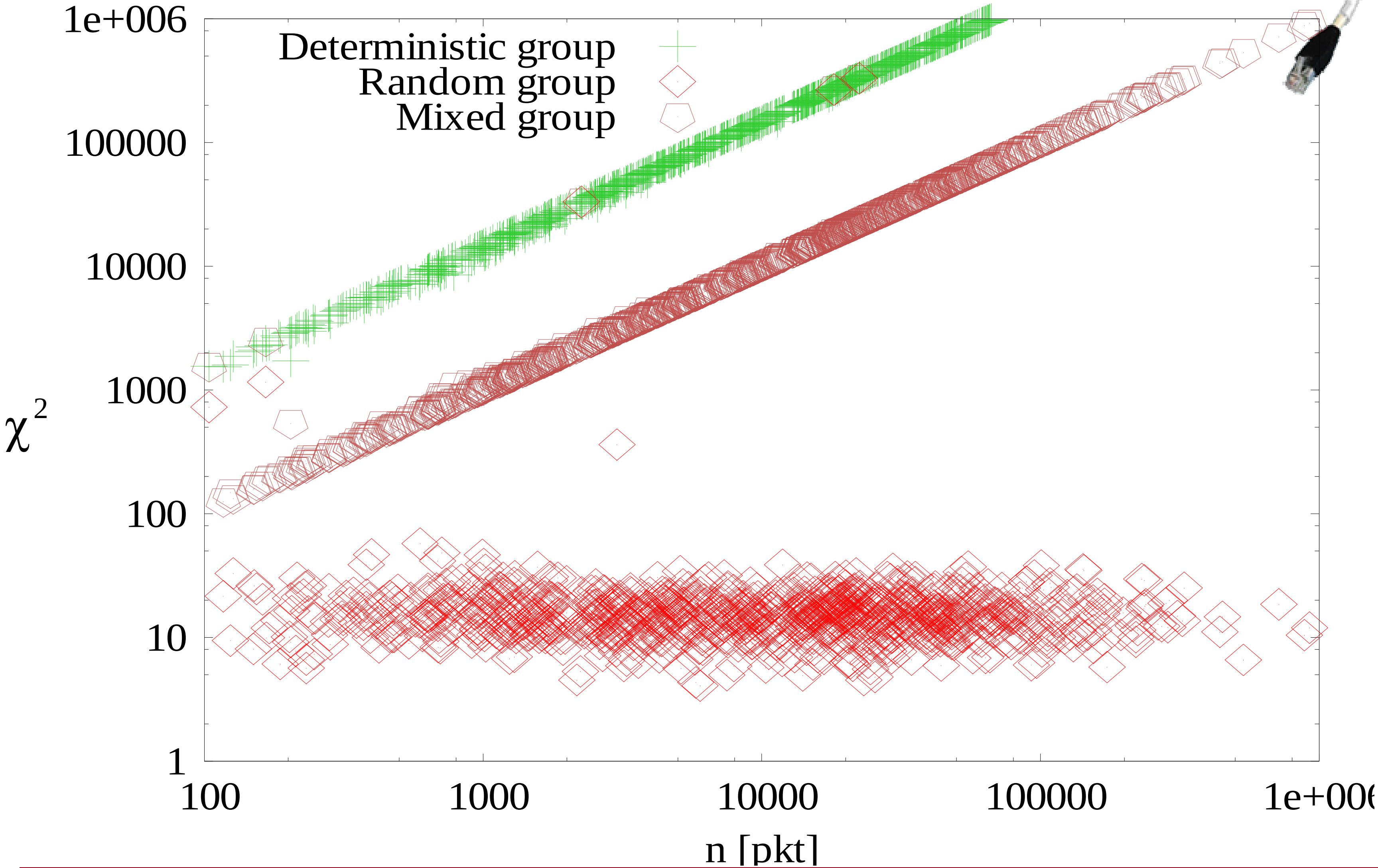
Question: Why comparing against a UNIFORM pdf??

$$\chi^2 = \sum_{i=1}^{2^b} \frac{(O_i - E_i)^2}{E_i}$$

4 bit long chunks: χ^2 evolution

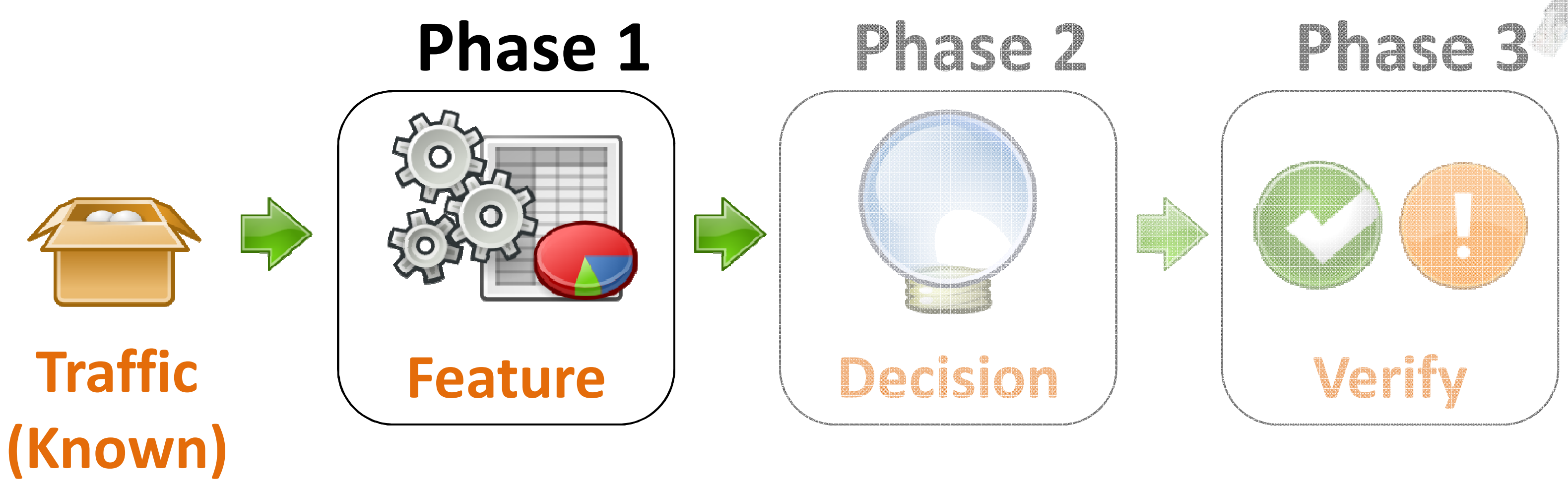


KISS



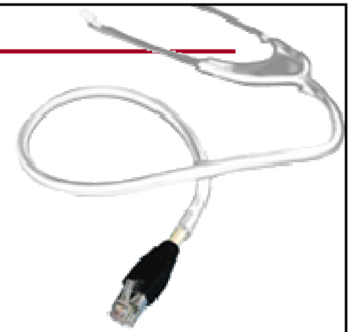
Deterministic group
Random group
Mixed group
1e+006
100000
10000
1000
100
10
1
χ^2
100
1000
10000
100000
1e+006
n [pkt]

Our Approach



- Statistical characterization of bits in a flow
  - Test χ^2
- Decision process
  - Minimum distance / maximum likelihood

C-dimension space



$$[\chi^2_1, \ldots, \chi^2_C]$$

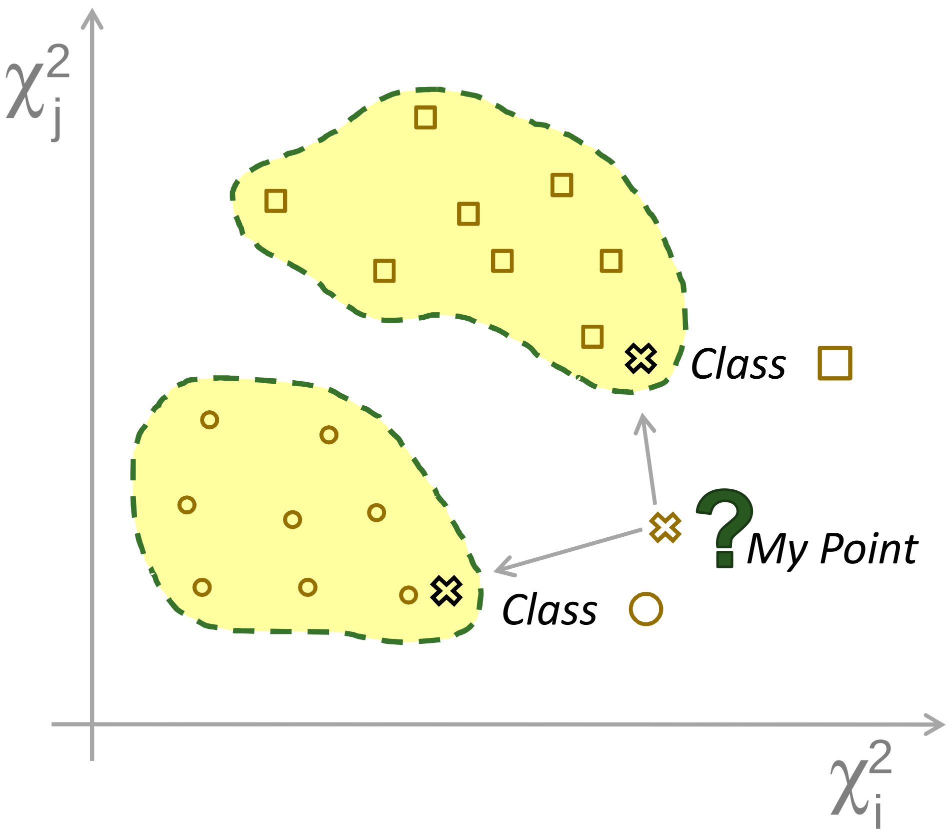
↓

Iperspace

↓

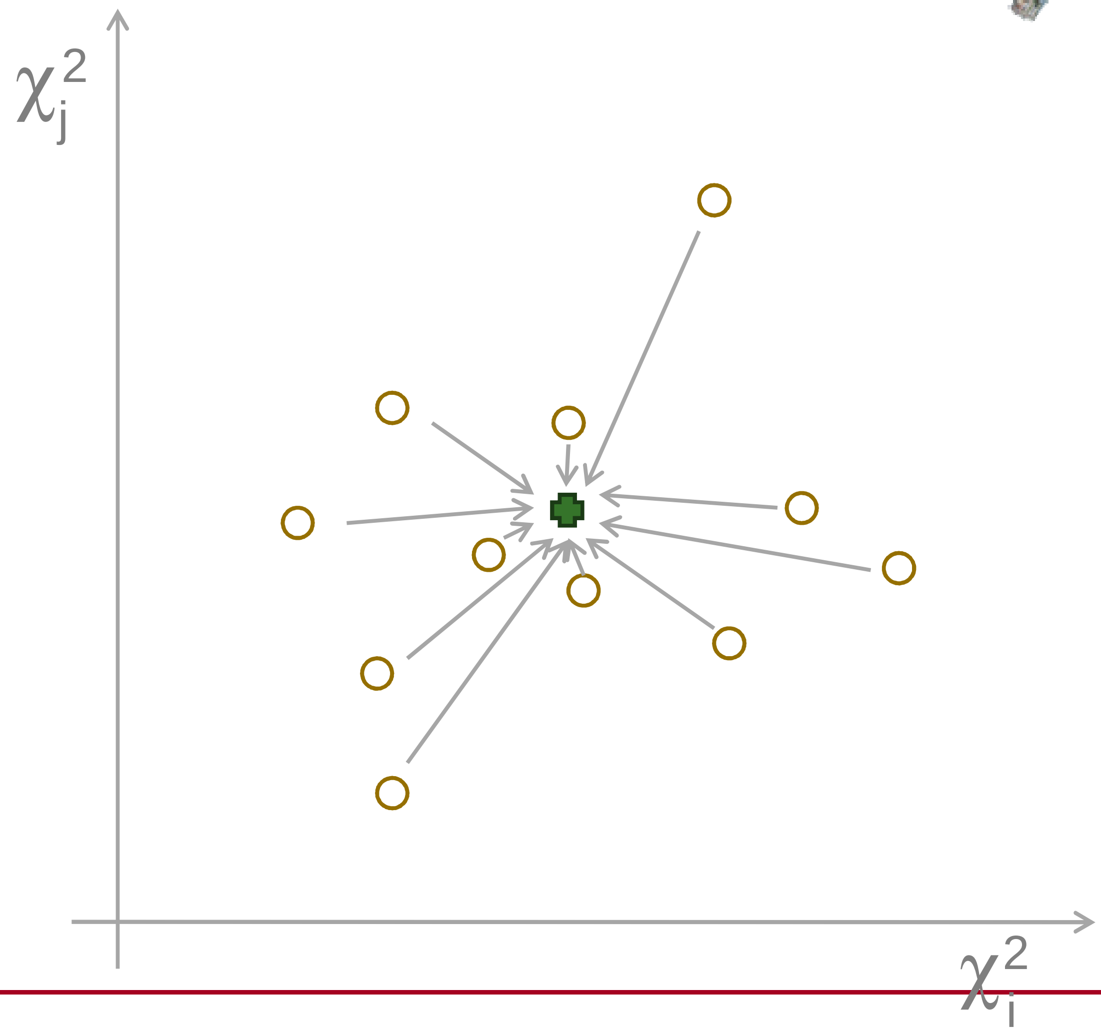
Classification Regions

Euclidean Distance | **Support Vector Machine**



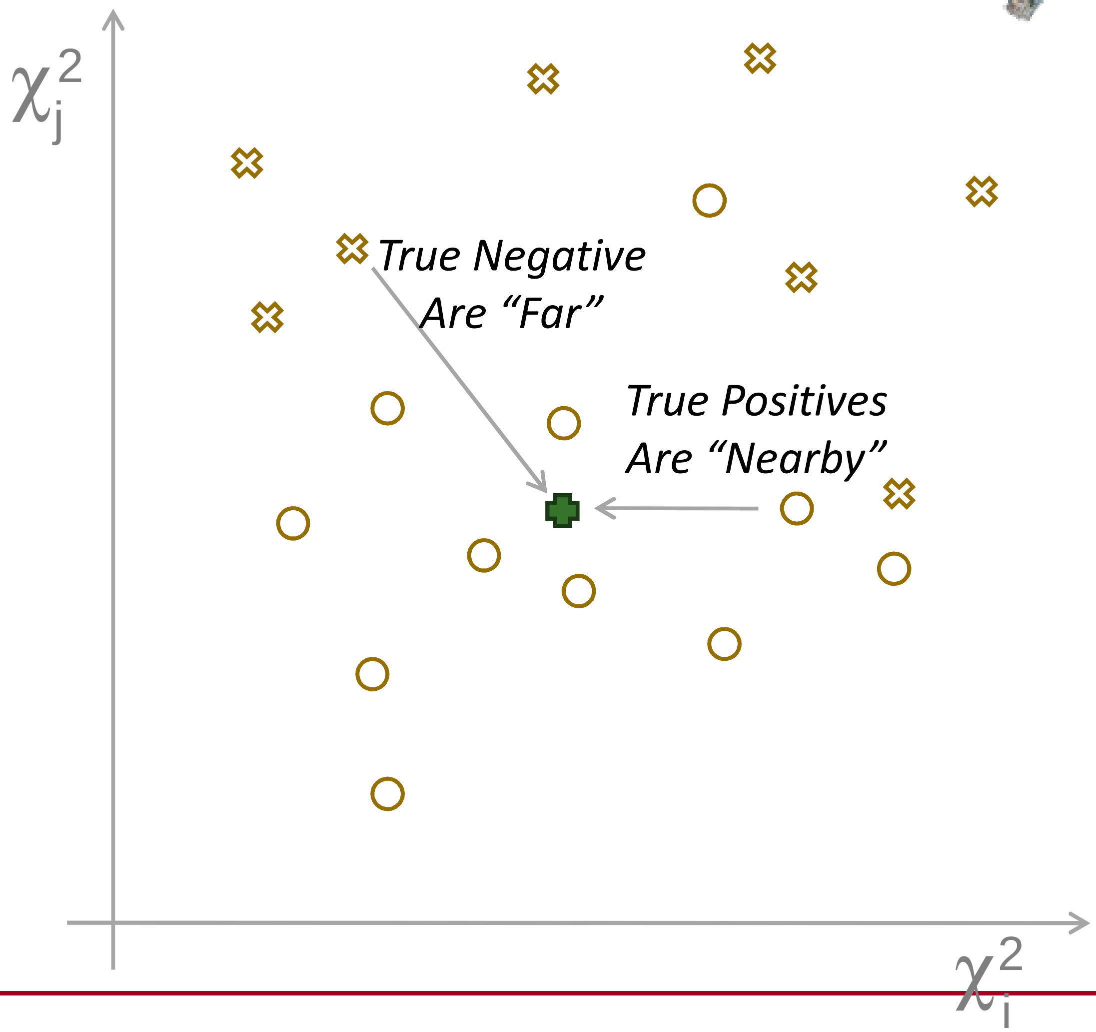
Euclidean Distance Classifier

- Centroid
  - Center of mass



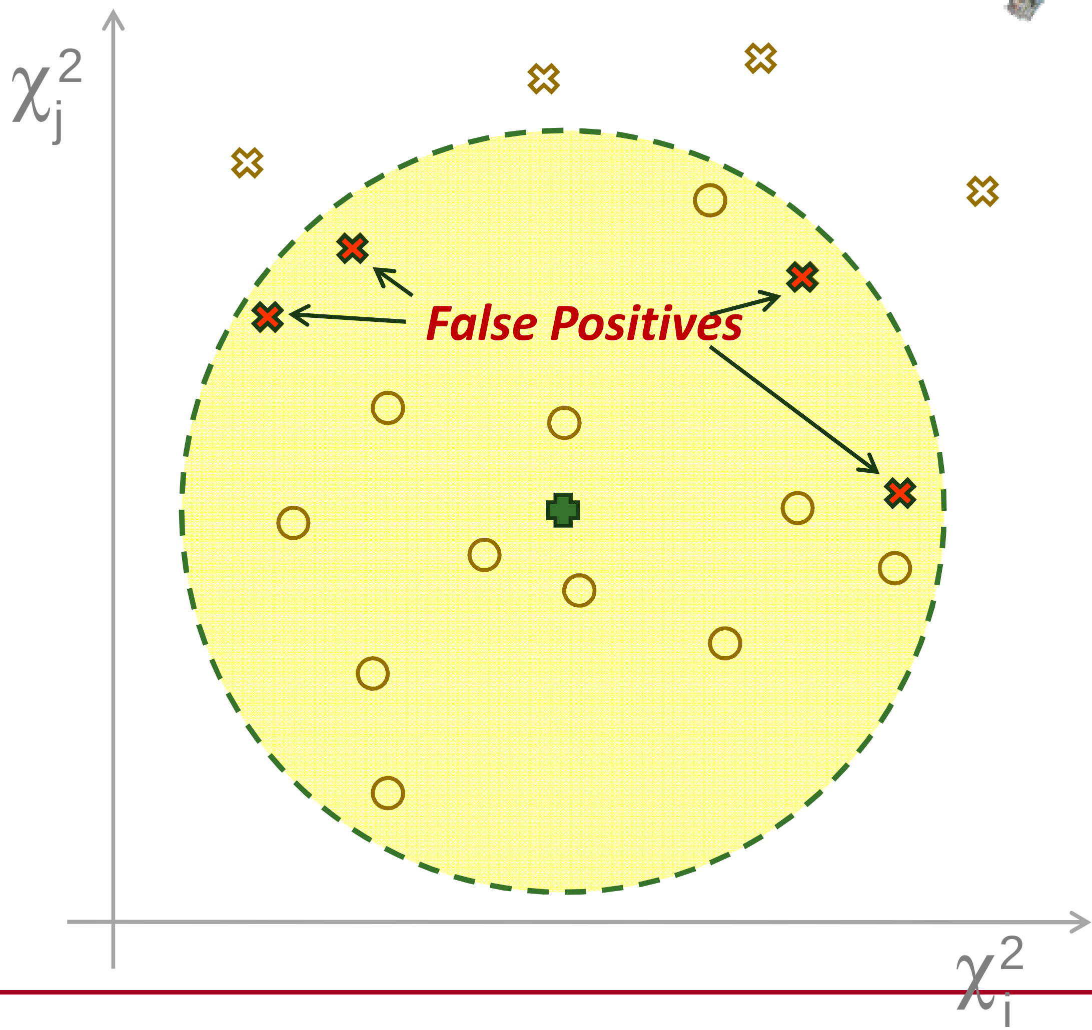
Euclidean Distance Classifier

- Centroid
  - Center of mass



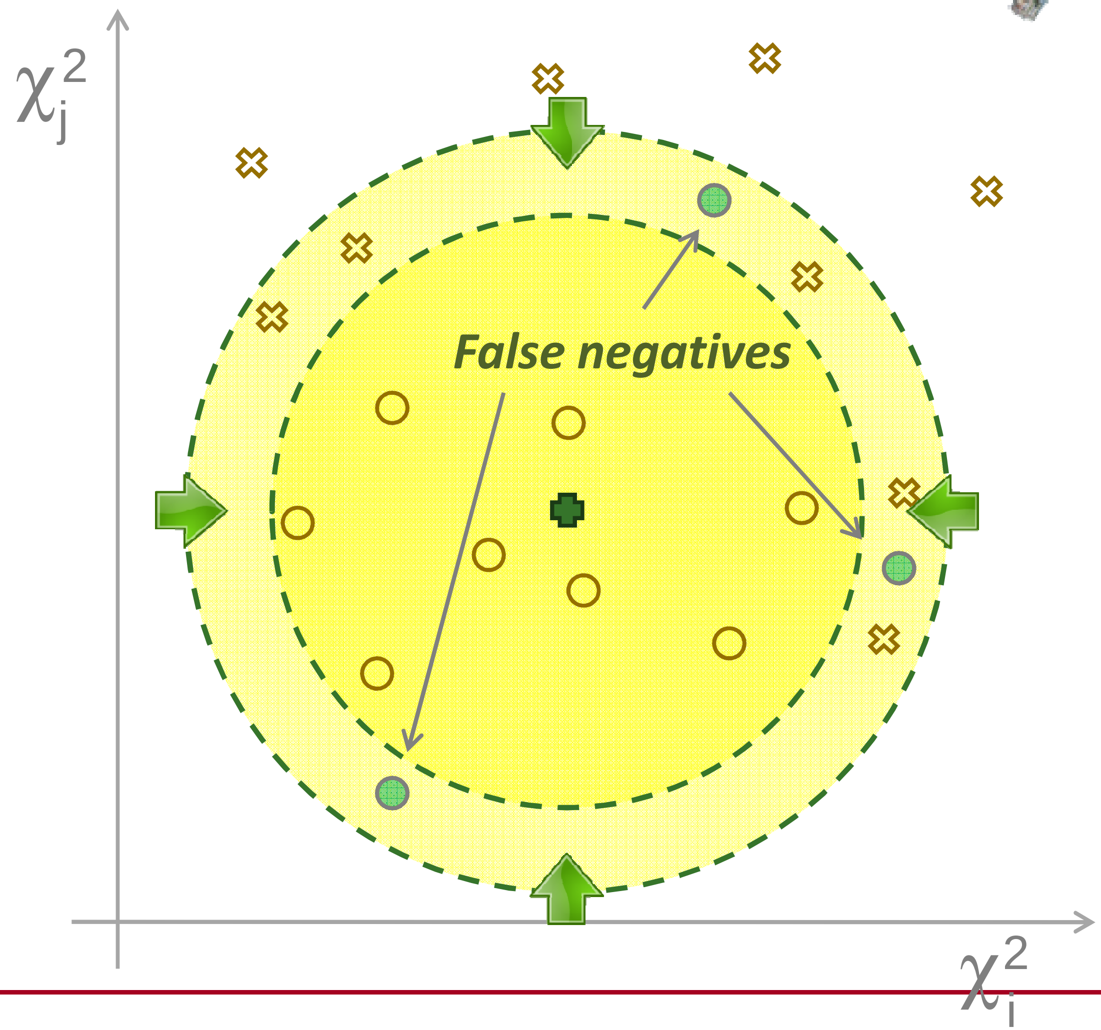
Euclidean Distance Classifier

- Centroid
  - Center of mass
- Iper-sphere



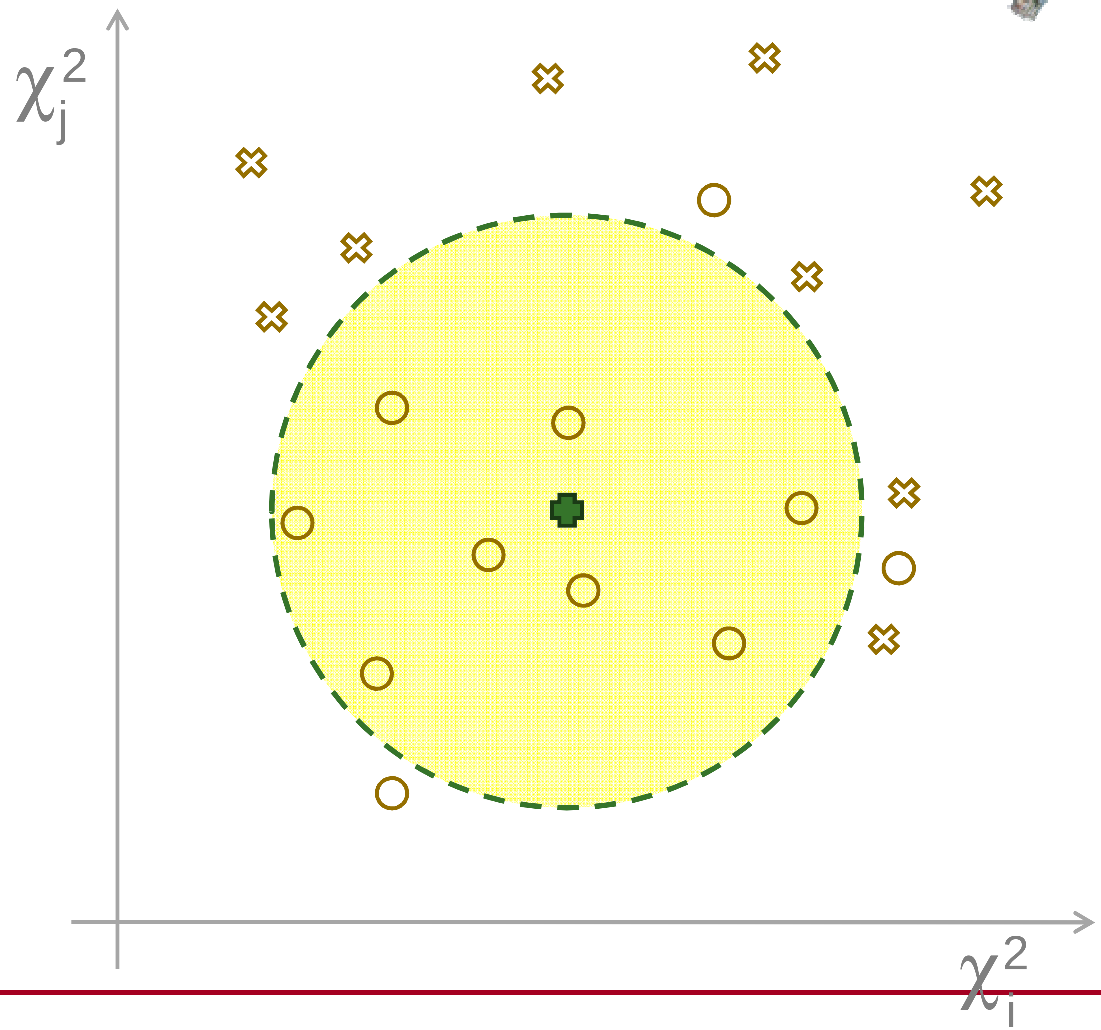
Euclidean Distance Classifier

- Centroid
  - Center of mass
- Iper-sphere
  - Radius



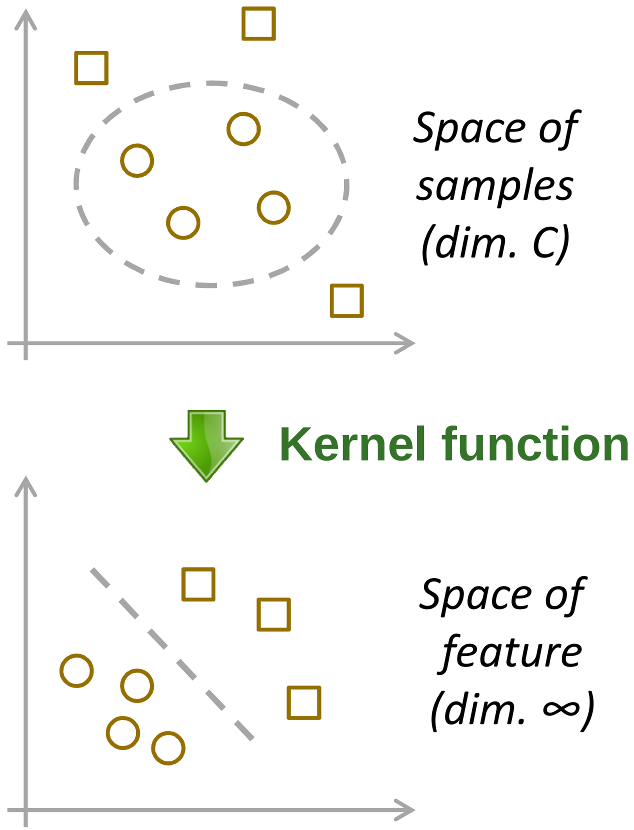
Euclidean Distance Classifier

- Centroid
  - Center of mass
- Iper-sphere
  - max { True Pos. }
  - min { False Neg. }
- Confidence
  - The distance is a measure of the condifence of the decision



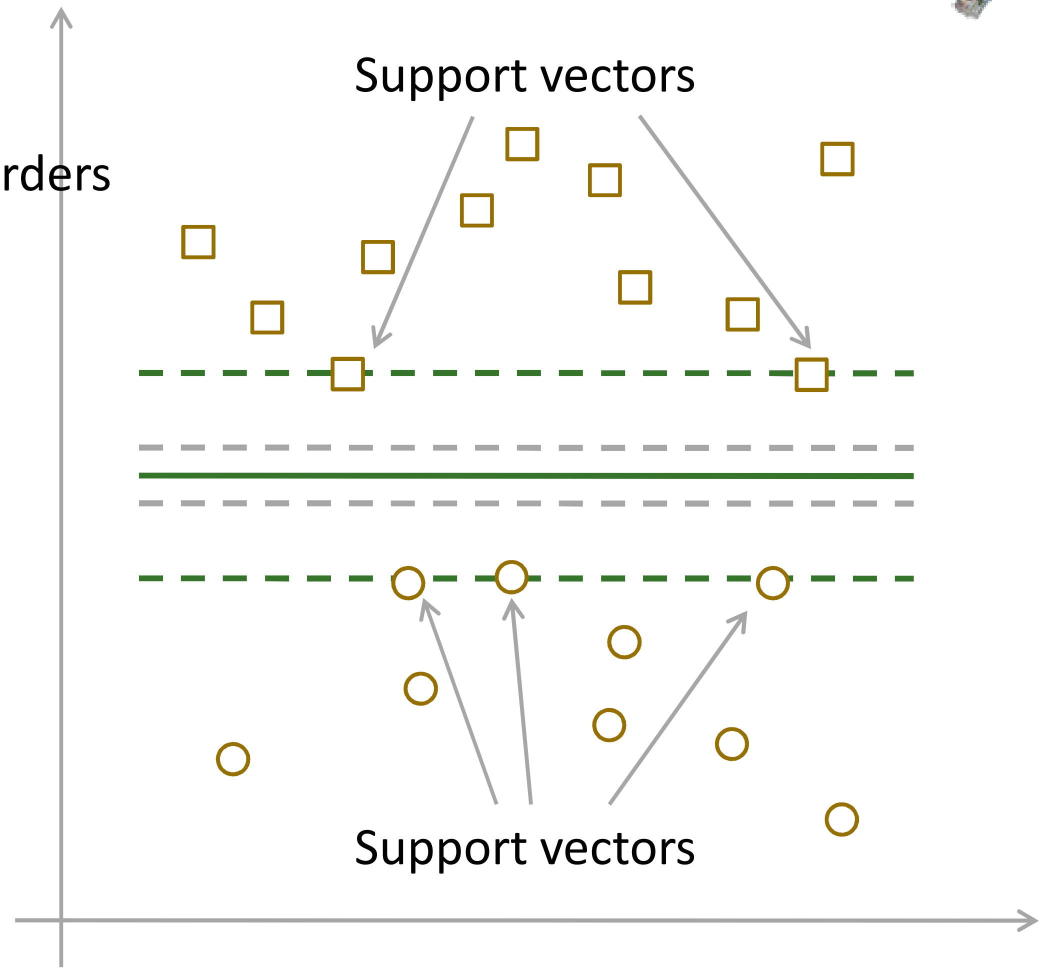
Support Vector Machine

- Kernel functions
  - Move point so that borders are simple



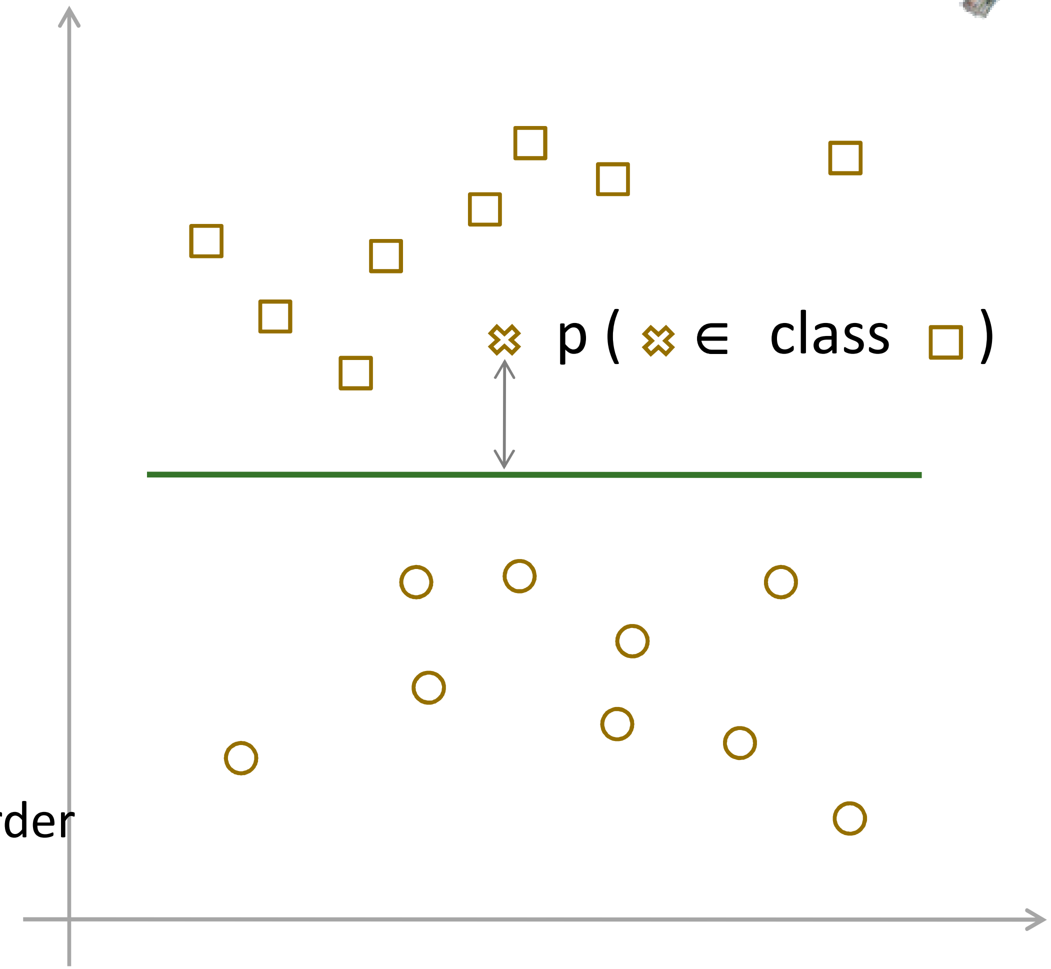
Support Vector Machine

- Kernel functions
  - Move point so that borders are simple
- Borders are planes
  - Simple surface!
  - Nice math
  - Support Vectors
  - **LibSVM**

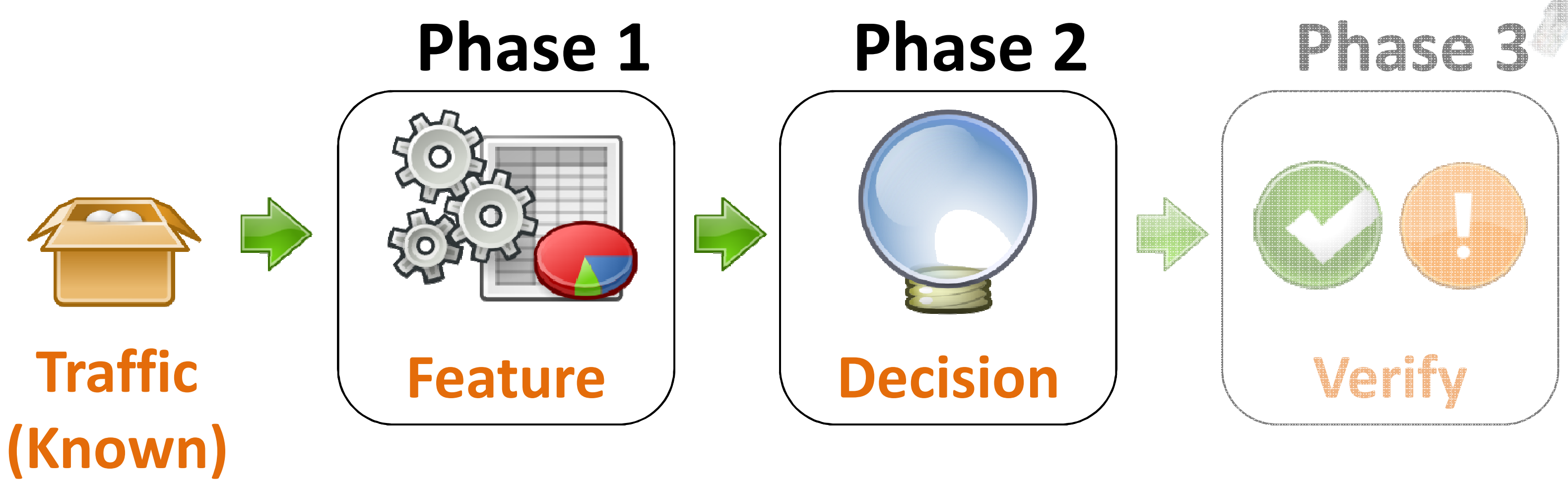


Support Vector Machine

- Kernel functions
- Borders are planes
  - Simple surface!
  - Nice math
  - Support Vectors
  - **LibSVM**
- Decision
  - Distance from the border
  - Confidence is a probability



Our Approach



- Statistical characterization of bits in a flow
  - Test χ^2
- Decision process
  - Minimum distance / maximum likelihood
- Performance evaluation
  - How accurate is all this?

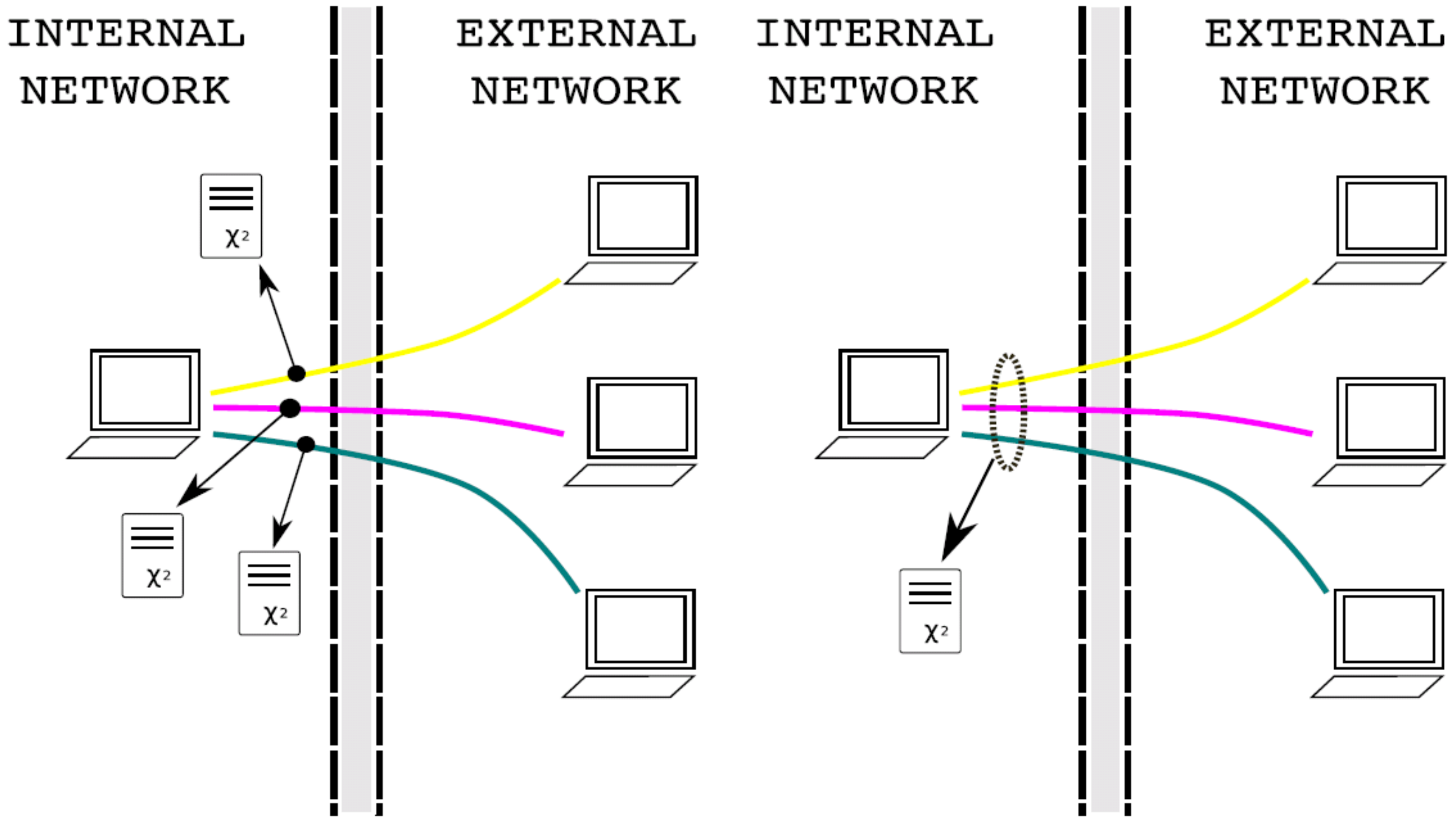
Per flow and per endpoint

- What are we going to classify?
  - It can be applied to both single flows
  - And to endpoints
- **Question:**
  - **Do we assume to monitor ALL packets?**
  - **Do we assume to monitor since the first packet?**

Per flow and per endpoint

- What are we going to classify?
  - It can be applied to both single flows
  - And to endpoints
- **Question:**
  - **Do we assume to monitor ALL packets?**
  - **Do we assume to monitor since the FIRST packet?**
- NO!
  - It is robust to sampling
  - It can start from any point

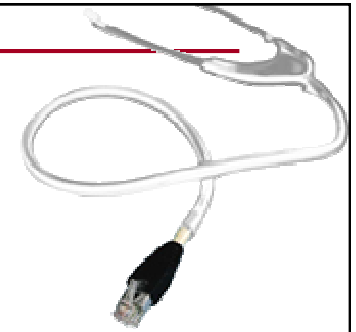
Per flow and per endpoint

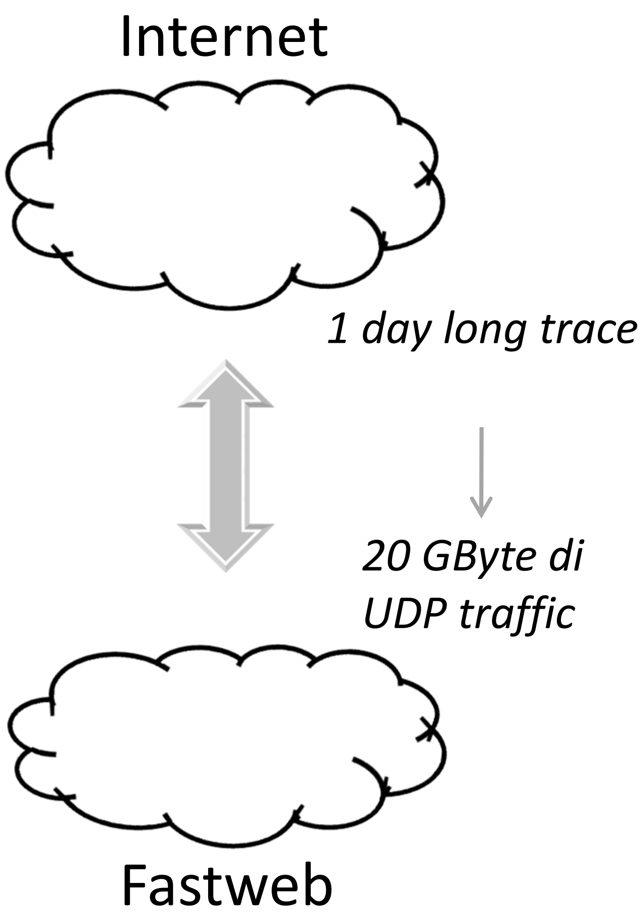


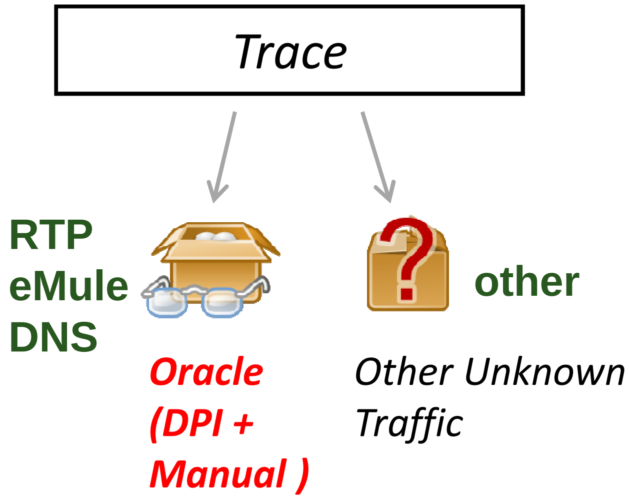
(a) Flow aggregation: a χ^2 signature is computed over every packets of a flow.

(b) Endpoint aggregation: a χ^2 signature is computed over every packets sent to (or received by) an endpoint.

Real traffic traces

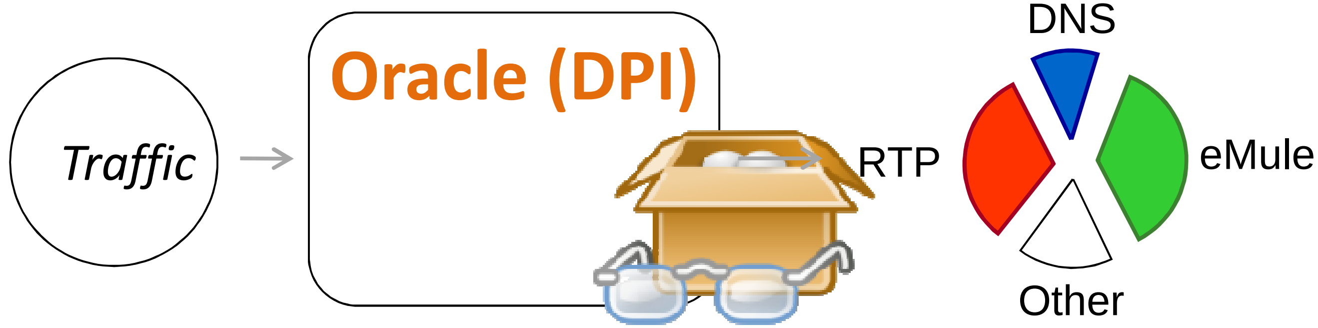




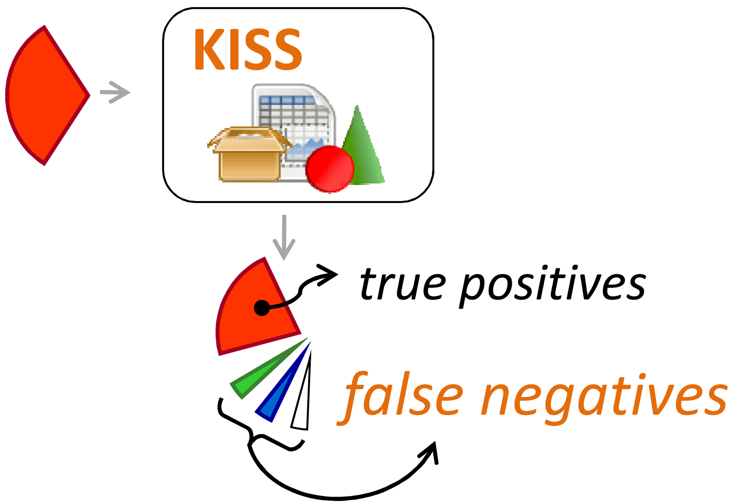


- Known + Other — **Training**
- Known Traffic — **False Negatives**
- Unknown traffic — **False Positives**

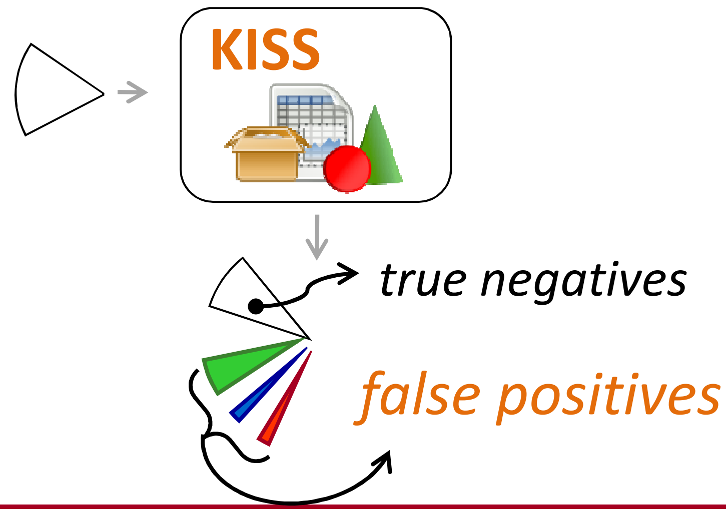
Definition of false positive/negative



Classifing "known"



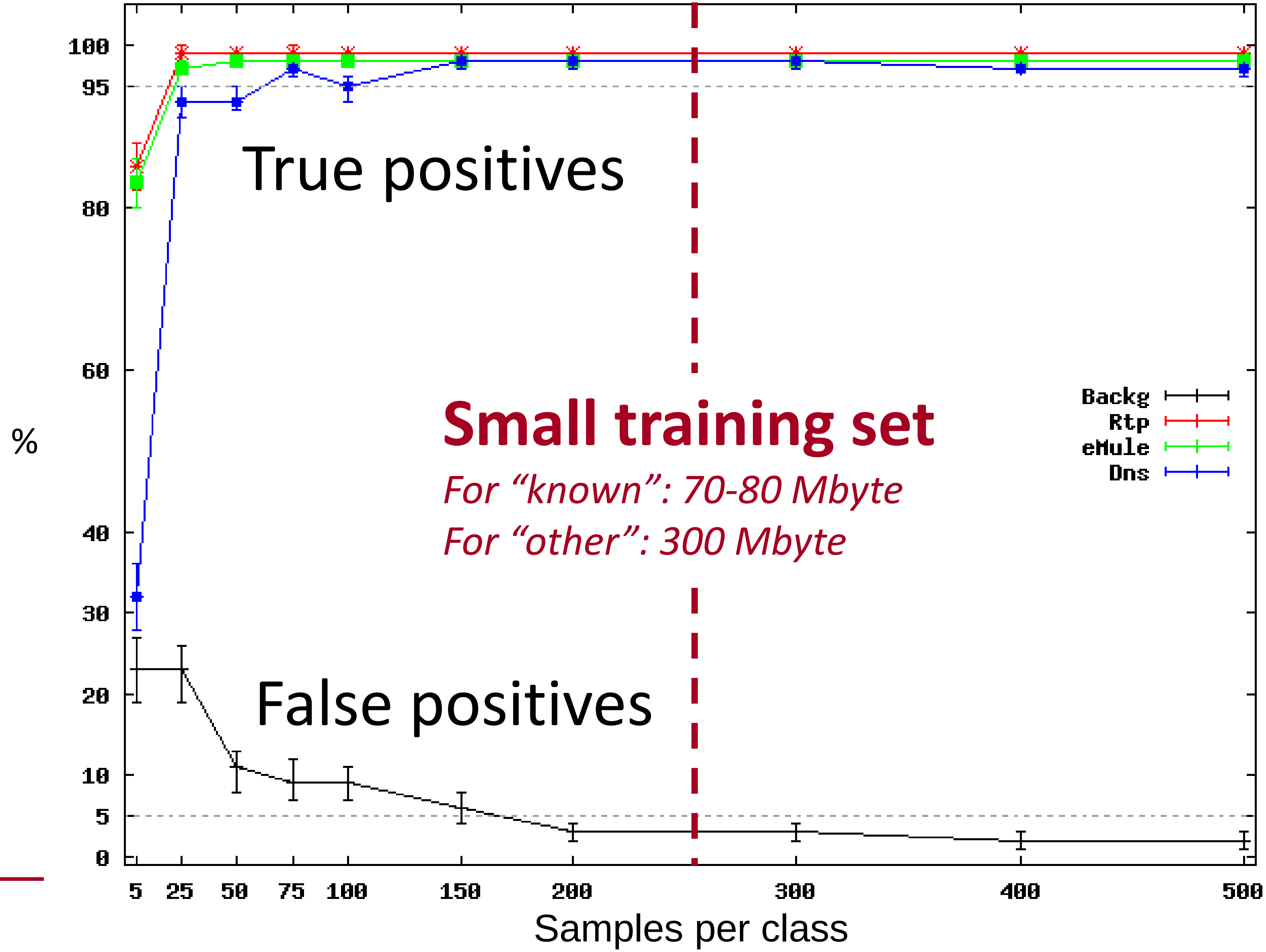
Classifing "other"



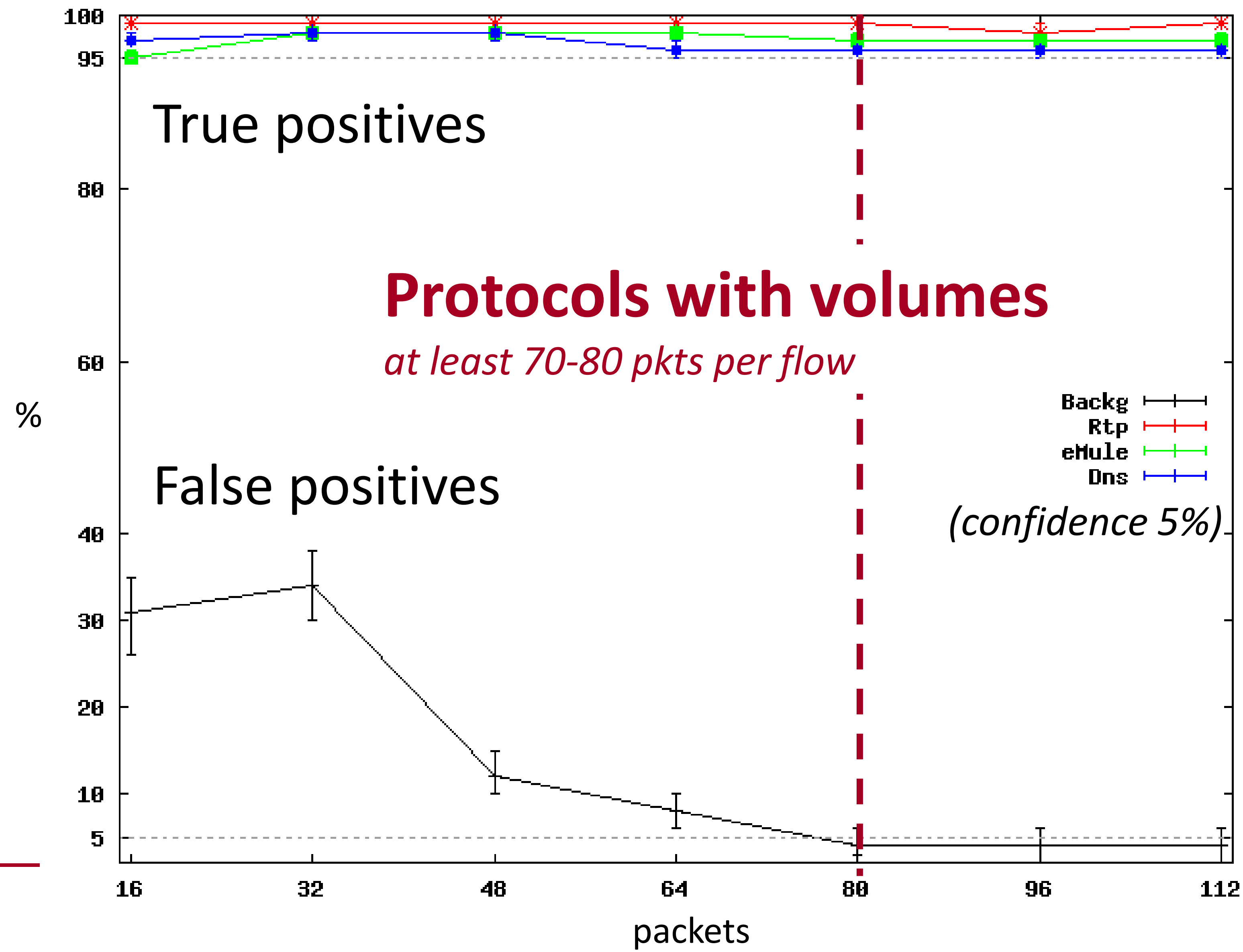
Results

		Euclidean Distance		SVM	
		Case A	Case B	Case A	Case B
Known traffic (False Neg.) [%]	Rtp	0.08	0.23	0.00	0.05
	Edk	13.03	7.97	0.98	0.54
	Dns	6.57	19.19	0.12	2.14
		Case A	Case B	Case A	Case B
Other (False Pos.) [%]	other	13.6	17.01	0.00	0.18

Tuning trainset size



Tuning Num. of Packets for χ^2



P2P-TV applications

- P2P-TV applications are becoming popular
- They heavly rely on UDP at the transport protocol
- They are based on proprietary protocols
- They are evolving over time very quickly
- How to identify them?
- ... After 6 hours, KISS give you results

	Tot.	Joost	PPLive	SopCast	TVants	Aggr.
Joost	33514	98.1	-	-	-	1.9
PPLive	84452	-	100.0	-	-	-
SopCast	84473	-	-	99.9	-	0.1
TVants	27184	-	-	-	100.0	-
Aggr.	1.2M	0.3	-	-	-	99.7

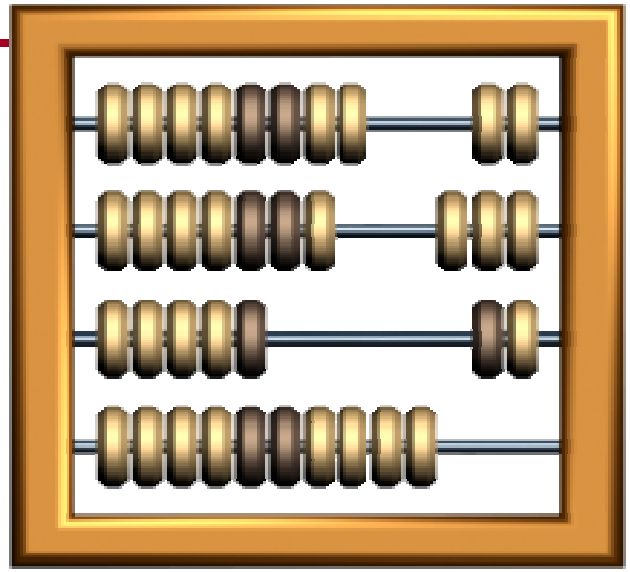
Putting all together

- Now with
  - 9 classes
  - 3 different networks

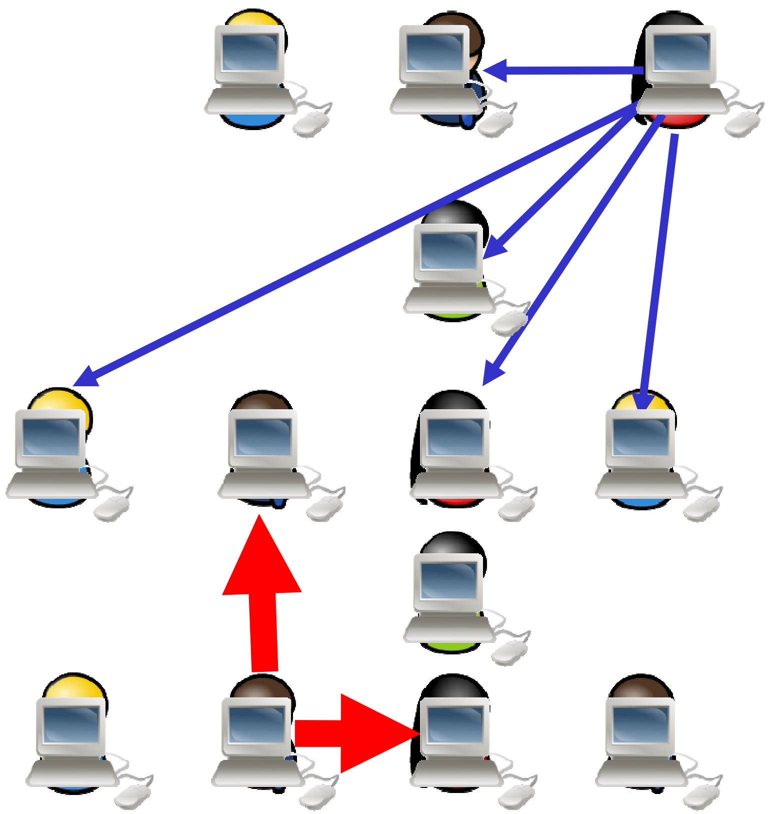
	Tot.	Bittorrent	Ed2k	RTCP	RTP	DNS	Skype	SopCast	TVAnts	PPLive	Backg
Bittorrent	1268	100.00	-	-	-	-	-	-	-	-	-
Ed2k	57255	0.02	95.97	-	-	0.03	3.16	-	-	-	0.80
RTCP	2407	-	-	99.96	-	-	-	-	-	-	0.04
RTP	585647	-	-	-	99.79	-	-	-	-	-	0.21
DNS	2707	0.46	-	-	-	99.54	-	-	-	-	-
Skype	46600	-	-	-	-	-	99.61	-	-	-	0.39
Sopcast	83460	-	-	-	-	-	-	99.95	-	-	0.05
TVAnts	25748	-	-	-	-	-	-	-	99.22	-	0.73
PPLive	27278	-	-	-	-	-	-	-	-	99.24	0.76
Backg	84273	0.27	7.59	0.01	2.67	0.22	-	-	-	-	89.21

Another example of behavioral classifier

Abacus



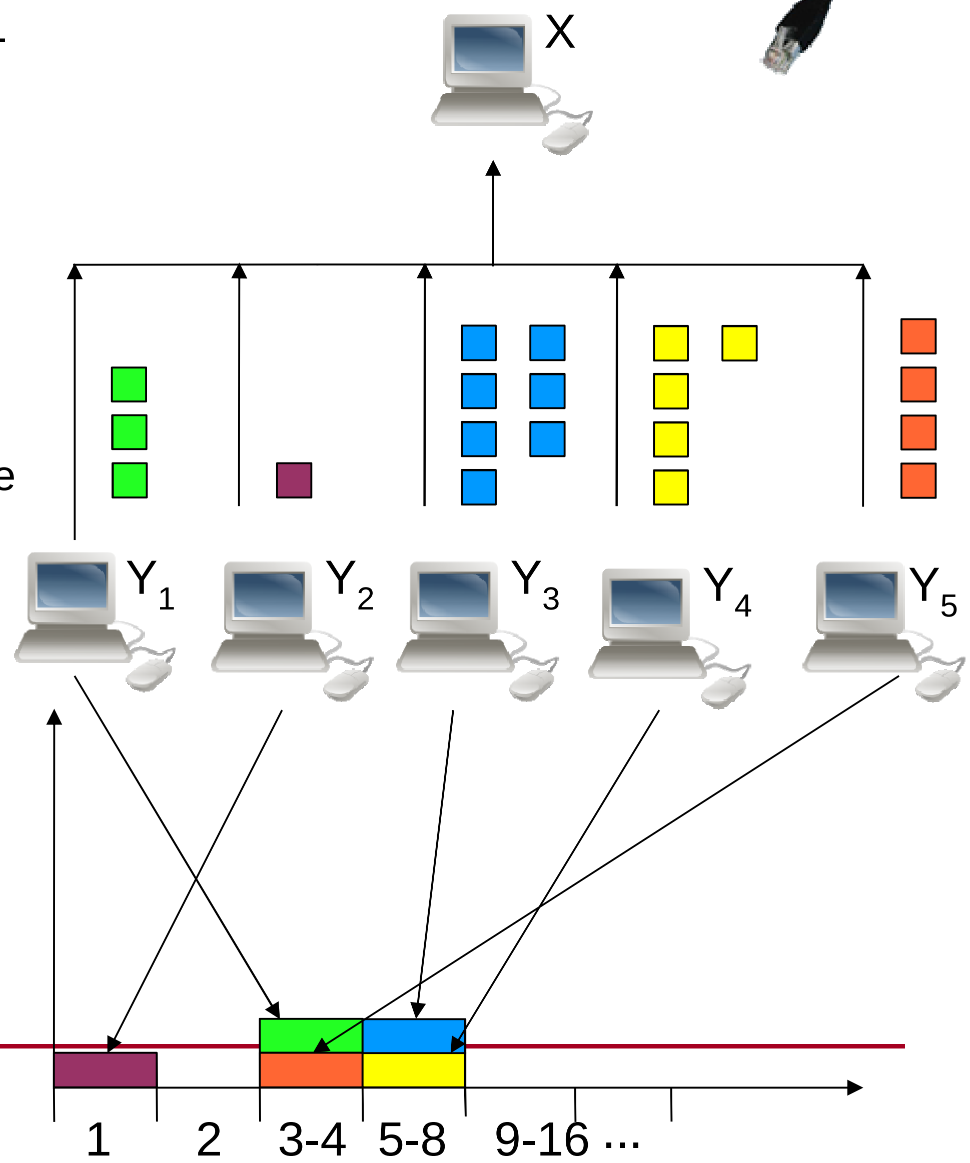
Abacus: Rationale



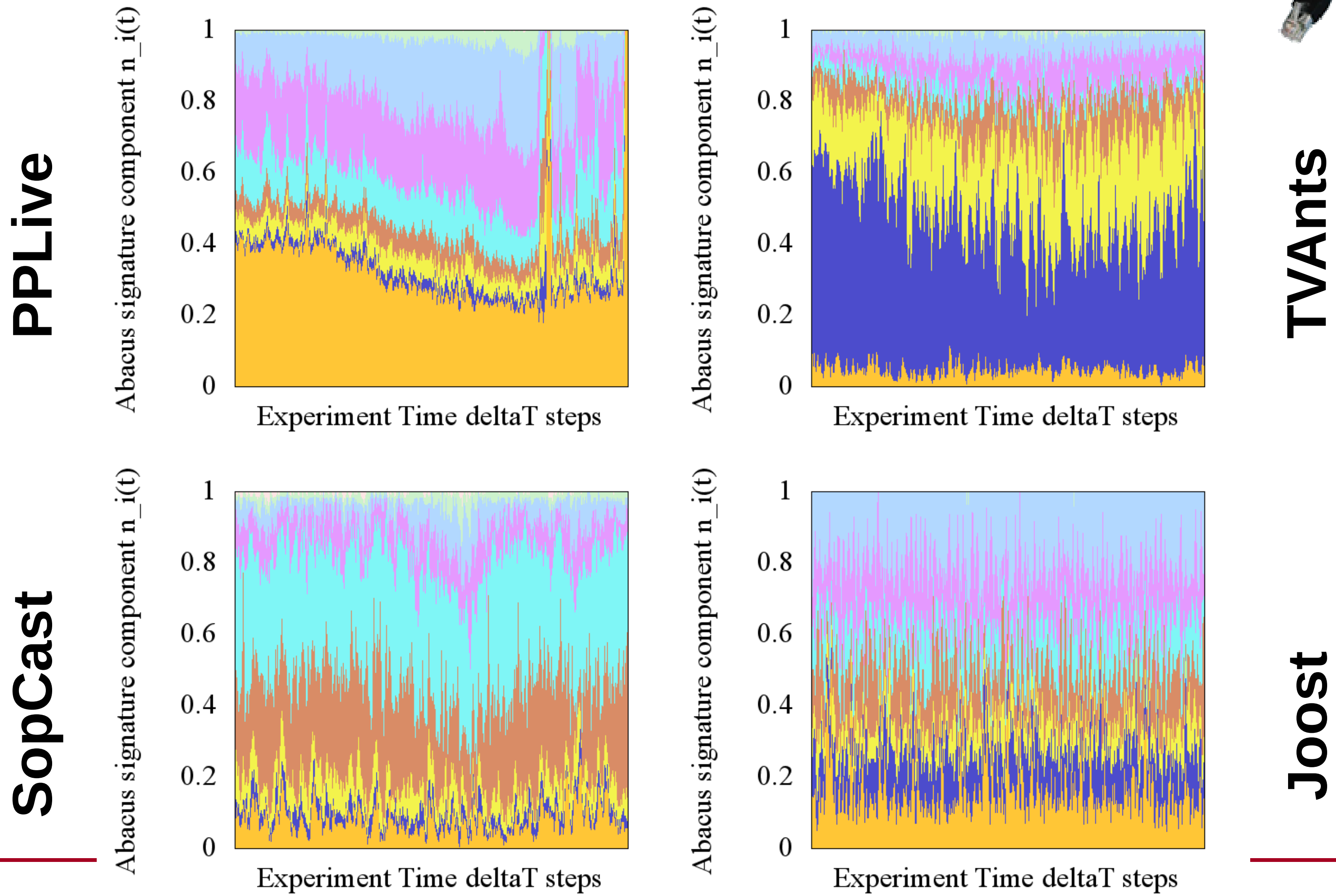
- Applications are like people in a party room
  - Some prefer brief exchanges with many other people
  - Some likes long talks with few other people
- “Attitudes” are different across P2P applications...
  - Some prefer to download small pieces of data from many peers
  - Some prefer to download all data from almost the same peers
- ... enough to classify them
  - Observe a host for a given time
  - Count the number of peers contacted and the number of packets exchanged which represent the *attitude*

Abacus signature definition

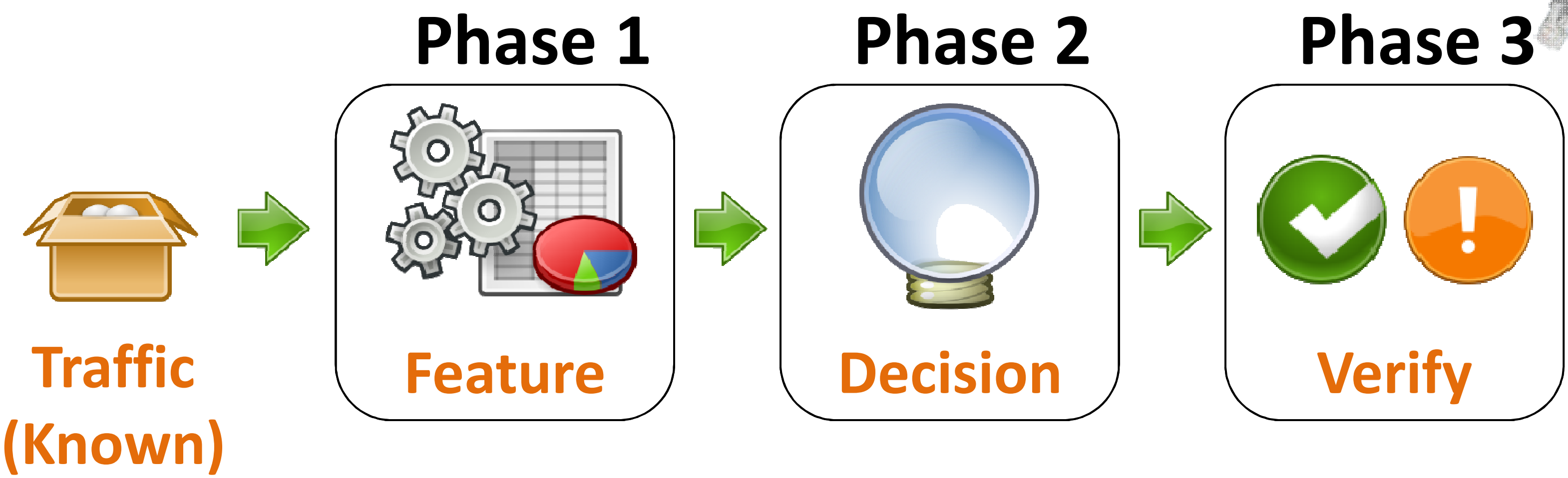
- Consider a host X which in a fixed time-window *ΔT = 5s* is contacted by N=5 peers Y_i
- for each peer Y_i count the number of packets sent to X in *ΔT*
- Consider a set of bins of exponential width
- Divide the peers in bins according to the number of exchanged packets
- Normalize the bins, i.e. divide for the total number of peers N
- The final signature is an *empirical probability distribution function*
- In the example
  - N=5, bins = (1, 0, 2, 2)☑
  - Abacus signature (0.2, 0, 0.4, 0.4)☑



Signature comparison



The Approach



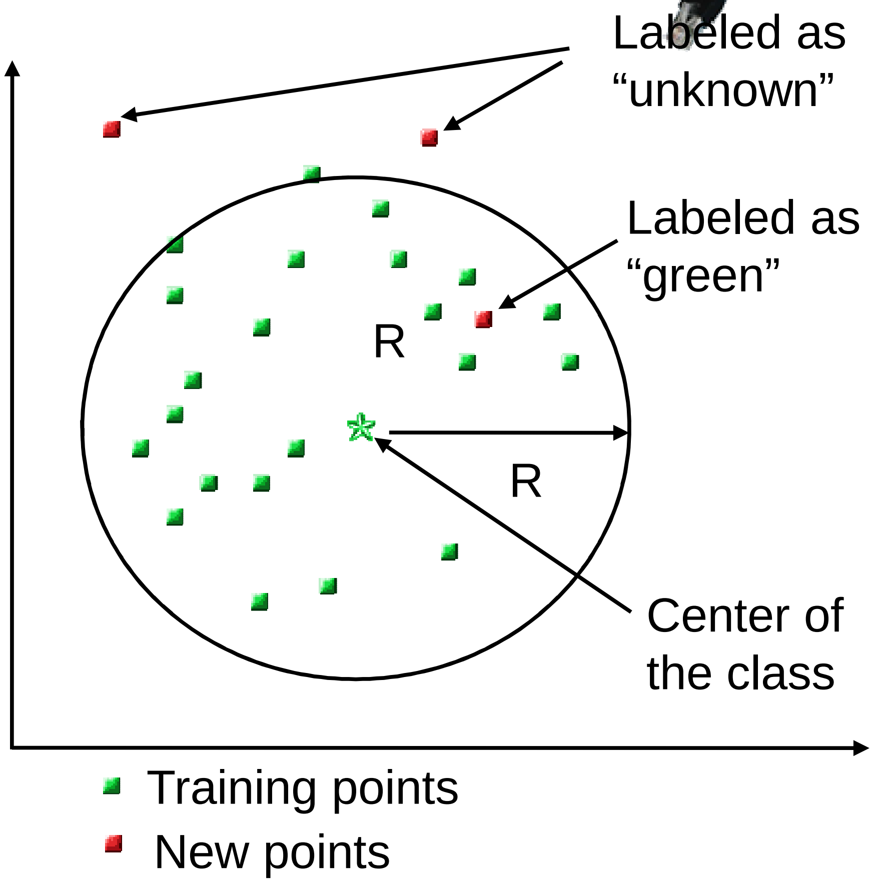
- Statistical characterization of bits in a flow
  - ABACUS signatures
- Decision process
  - Supervised machine learning based on SVM
- Performance evaluation
  - How accurate is all this?

Experimental results

		Abacus signature confusion matrix				
		Classification outcome				
		PPLive	TVAnts	SopCast	Joost	Unk
Real value	PPLive	81.66	0.58	9.55	2.32	5.9
	TVAnts	0.41	98.84	0.15	0.57	0
	SopCast	3.76	0.11	89.62	0.32	6.2
	Joost	2.84	0.55	0.28	89.5	6.9

Rejection criterion

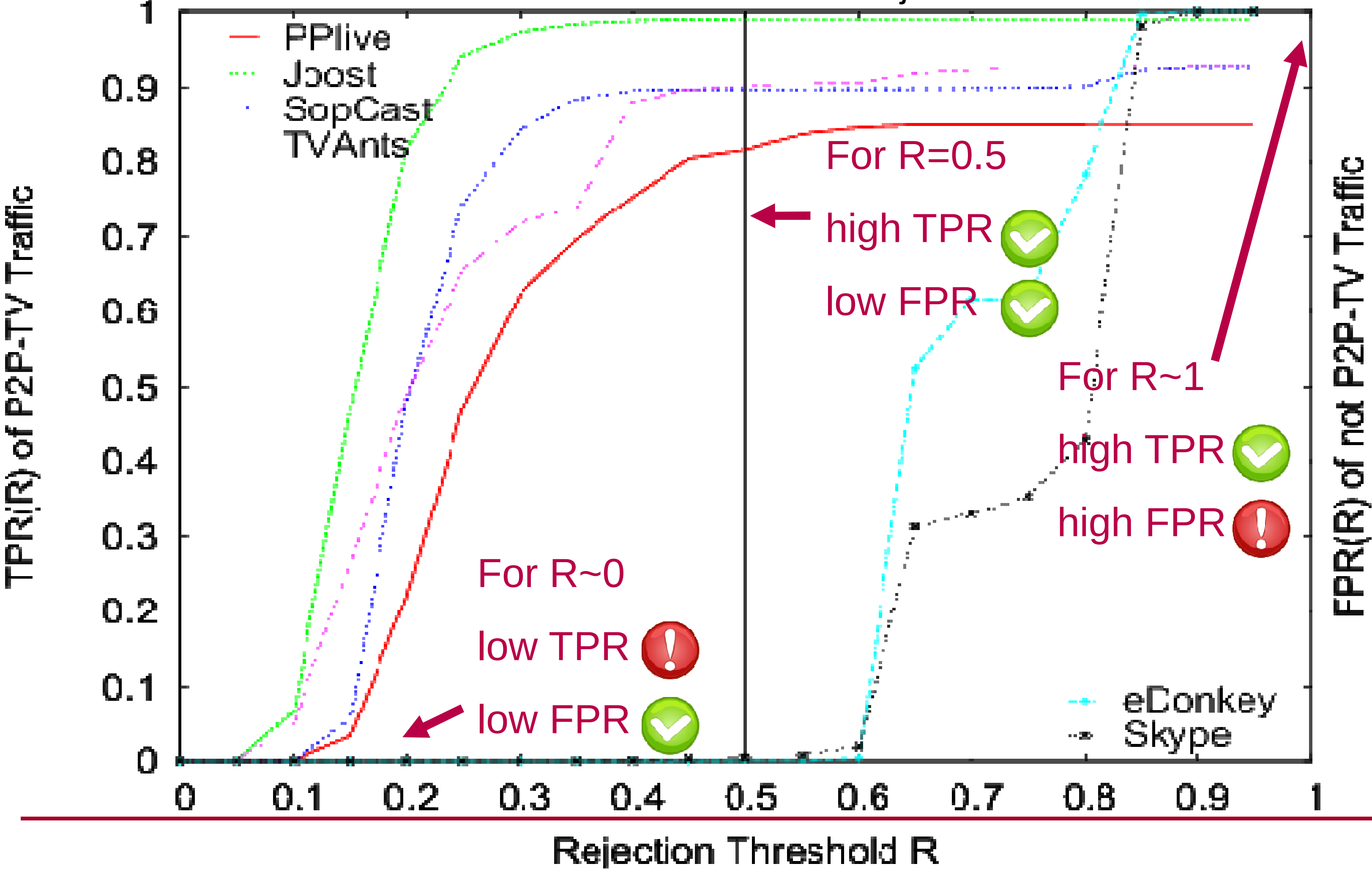
- **Hyper-space is partitioned**
  - **every point is given a label**
  - **even “unknown” apps**
- Need a way to recognize them
  - Define a center for each class
  - Define a threshold *R*
  - Calculate the distance *d* between the point and the center of the assigned class
  - If d > R mark the new point as unknown
- Bhattacharyya distance BD
  - Distance between p.d.f. $BD(p,q) = \sqrt{1-B}$



$$B = \sum_{x=1}^{n} \sqrt{p(x) * q(x)}$$

Experimental results

For “unknown traffic” the selection of the rejection threshold R is fundamental!



Creating Communication Technologies

Automatic Traffic Classification

Semi-supervised learning approach

Machine-Learning (ML) in TRAC

ML was introduced to enhance port/payload-based traffic classification:

Supervised ML: based on what I ALREADY KNOW

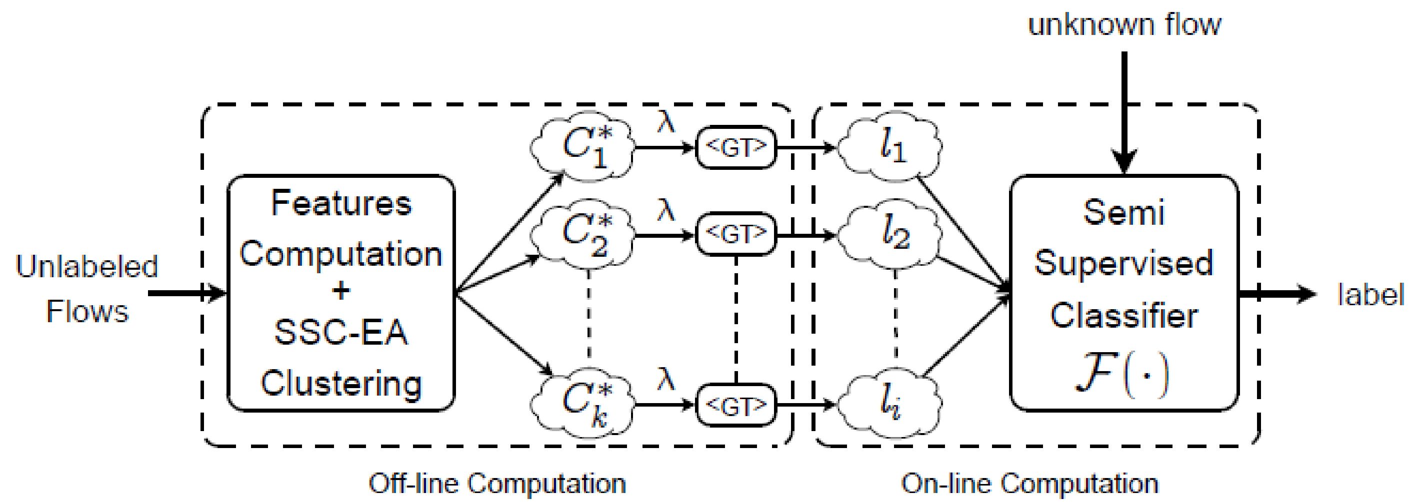
- (+) improves traditional classification techniques.
- (−) needs training on full-labeled traffic datasets.
- (−) labeling traffic flows is difficult, time-consuming, and costly.

Unsupervised ML: KNOWLEDGE-INDEPENDENT analysis

- (+) **Clustering**: separate flows in classes sharing similar characteristics.
- (+) classification is done by limited labeled traffic (**Semi-Supervised ML**).
- (−) lack of robustness: general clustering algorithms are sensitive to initialization, specification of number of clusters, etc.
- (−) difficult to cluster high-dimensional data: structure-masking by irrelevant features, sparse spaces (“the curse of dimensionality”).

Machine Learning in TRAC: our Proposal

We want to reduce the need of labeled traffic, limiting the impacts on classification accuracy.



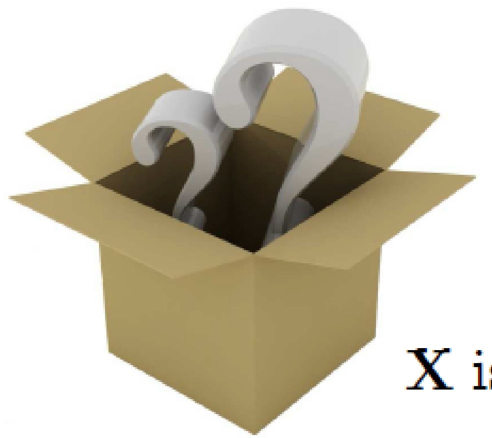
- Two-steps approach: Clustering + Semi-Supervised Classification.

Robust Clustering on unlabeled traffic flows: enhance clustering through the combination of Sub-Space Clustering + Evidence Accumulation.

- Label Clusters: use a small fraction λ of labeled flows per cluster.
- Distance-based Classification: assign closest-cluster's label.

Clustering for Traffic Analysis (Off-line)

- Let $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be a set of n flows captured at the network of analysis.
- Each flow $\mathbf{y}_i \in \mathbf{Y}$ is described by a set of m traffic features: $\mathbf{x}_i = (x_i(1), .., x_i(m)) \in \mathbb{R}^m$.
- $\mathbf{X} = \{\mathbf{x}_1, .., \mathbf{x}_n\}$ is the complete matrix of features, referred to as the *feature space*.

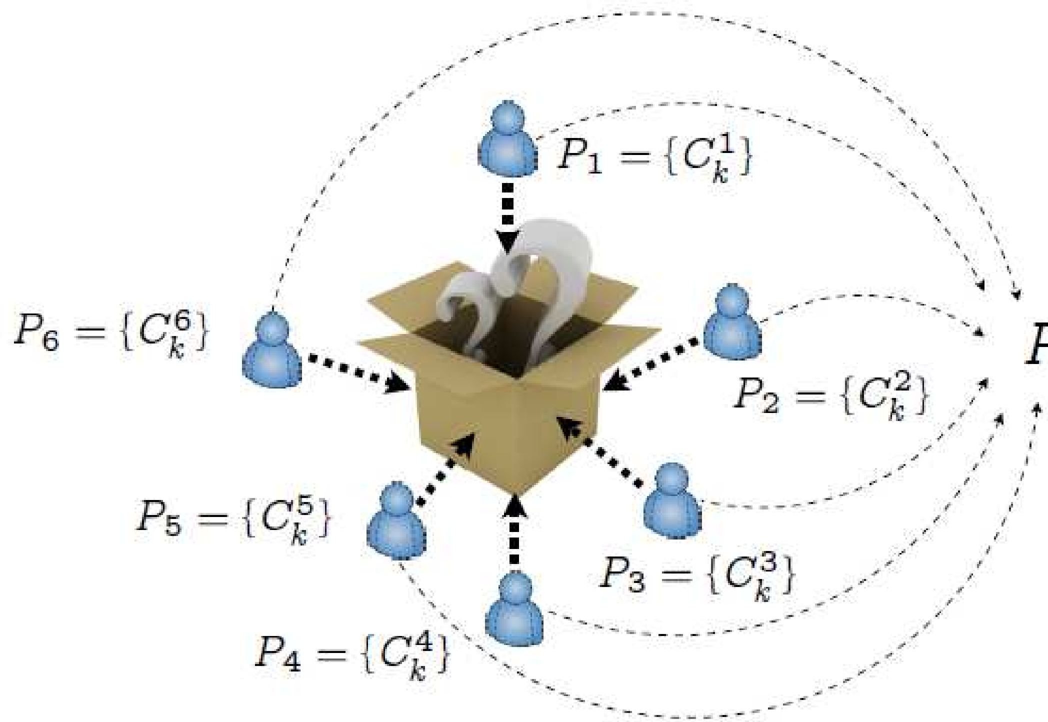


$\mathbf{X}$ is a black box

Retrieve natural groupings in $\mathbf{X}$ through clustering is challenging!!!

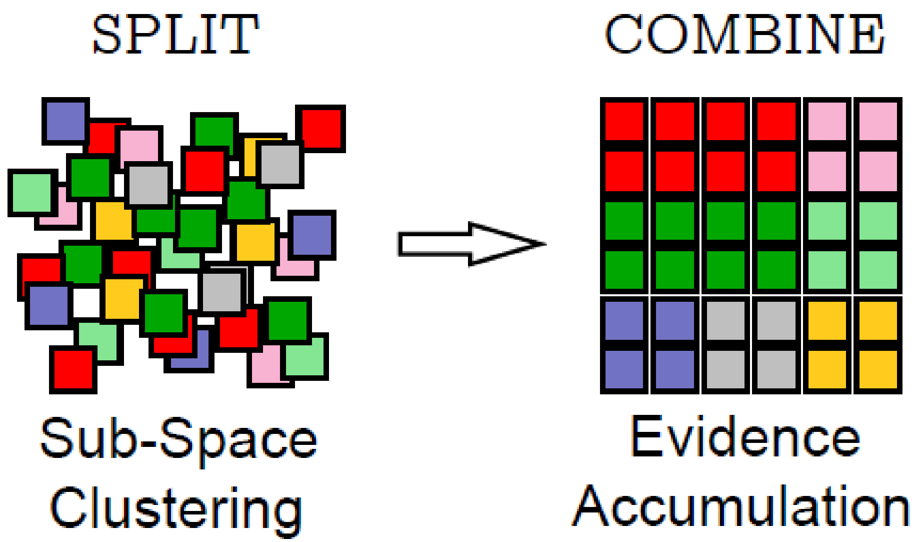
How to Improve Clustering Robustness?

- Idea: combine the information provided by multiple partitions of $\mathbf{X}$ to “filter noise”, easing the discovery of natural groupings.
- How to produce multiple partitions? $\rightarrow$ Sub-Space Clustering.
- Each sub-space $\mathbf{X}_i \subset \mathbf{X}$ is obtained by projecting $\mathbf{X}$ in k out of the m original dimensions. Density-based clustering (DBSCAN) at $\mathbf{X}_i$.



$P^* = \{C_k^*\} \rightarrow$ natural groupings

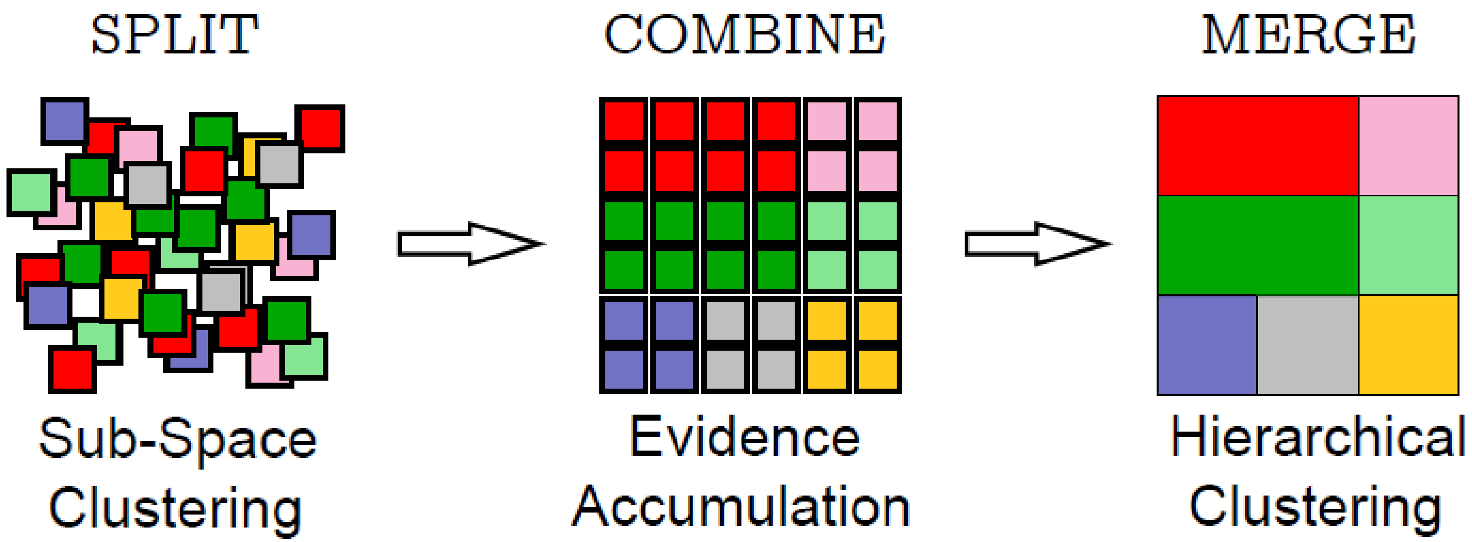
Evidence Accumulation to Retrieve Natural Groupings



Using Sub-Space Clustering we have SPLIT the problem, how do we COMBINE the obtained partitions? $\longrightarrow$ Evidence Accumulation

- Build a new inter-flows similarity measure S from the N partitions P_i.
- Flows belonging to a natural cluster C_k^* are likely to be co-located in the same cluster in different partitions P_i at different sub-spaces $\mathbf{X}_i$.
- $S(i,j) = n_{ij}/N$, where n_{ij} is the # of times that flows $\mathbf{y}_i$ and $\mathbf{y}_j$ were assigned to the same cluster through the N partitions.

Evidence Accumulation to Retrieve Natural Groupings

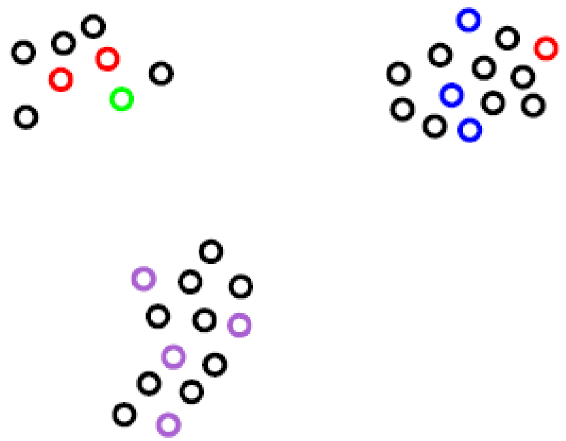


Using Sub-Space Clustering we have SPLIT the problem, how do we COMBINE the obtained partitions? $\longrightarrow$ Evidence Accumulation

The final partition $P^* = \{C_k^*\}$ is obtained by Hierarchical Clustering on S, MERGING the most similar flows into clusters C_k^*.

Semi-Supervised Classification

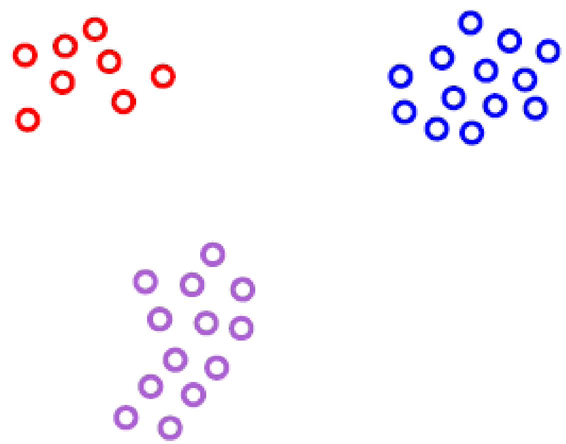
We build a classifier $\mathcal{F}(\cdot)$ from the obtained clusters:



- "Dig" the labels of a small fraction λ of flows (e.g., through DPI).

Semi-Supervised Classification

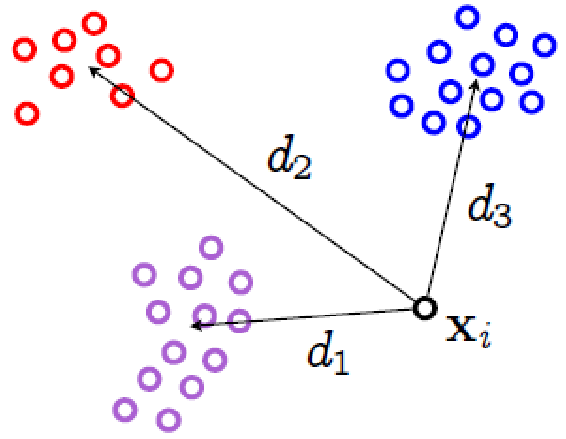
We build a classifier $\mathcal{F}(\cdot)$ from the obtained clusters:



- "Dig" the labels of a small fraction λ of flows (e.g., through DPI).
- Maximum-Likelihood Labeling: label each cluster with the most present label among the λ flows.

Semi-Supervised Classification

We build a classifier $\mathcal{F}(\cdot)$ from the obtained clusters:



- "Dig" the labels of a small fraction λ of flows (e.g., through DPI).
- Maximum-Likelihood Labeling: label each cluster with the most present label among the λ flows.
- Classify an unknown flow $\mathbf{y}_i$ based on its distance to the centroid of each cluster:

$$\text{label}_i = \mathcal{F}(\mathbf{x}_i) = \text{label}\left(\arg\min_k d(\mathbf{x}_i, \mathbf{o}_k^*)\right)$$

Traffic Datasets and Traffic Features

UNIBIS dataset (2000 flows)

- Controlled campus network traffic, labeled through GT classifier.
- 4 traffic classes: HTTP, eMail (SSL), P2P (BitTorrent, Edonkey), and VoIP (Skype) (500 flows per traffic class).

VALTC dataset (4000 flows)

- Controlled isolated network traffic, labeled through GT classifier.
- 8 traffic classes: HTTP, eMail (POP3), P2P (Emule, LimeWire, Azureus), VoIP (Skype), monitoring traffic, file hosting/download.

Standard 22 Traffic Features

- proto, flow duration, flow volume (bytes and pkts), pkt length (min, mean, max, dev), and inter-arrival time (min, mean, max, dev).
- features are computed in both directions.

SSC-EA vs DBSCAN vs k-means

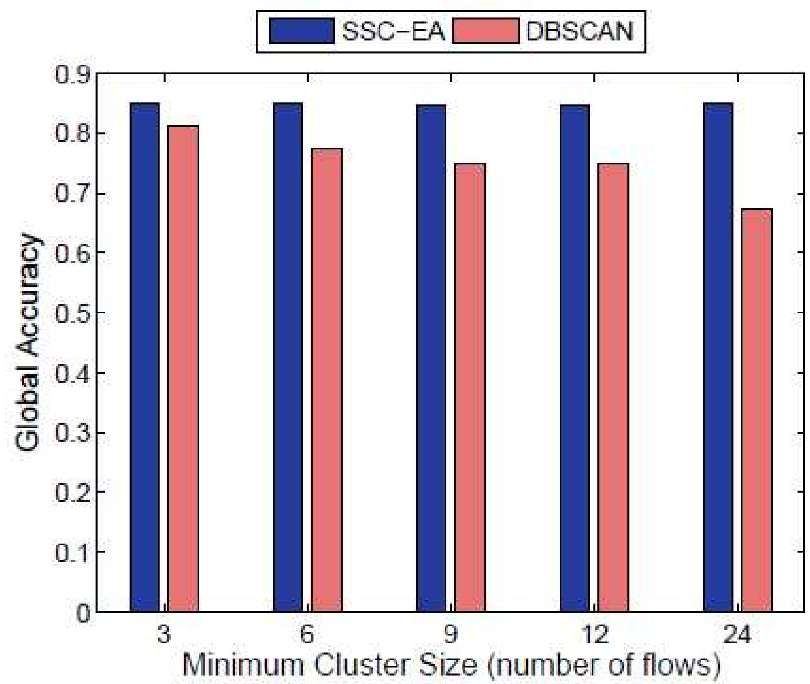
We measure clustering performance through Global Accuracy (GA) and Average per-Cluster Homogeneity (ACH):

$$\boxed{\mathrm{GA} = \frac{\sum_{k=1}^{n_{\mathrm{cls}}} TP(k)}{n}, \quad \mathrm{ACH} = \frac{\sum_{k=1}^{n_{\mathrm{cls}}} \frac{TP(k)}{n(k)}}{n_{\mathrm{cls}}}}$$

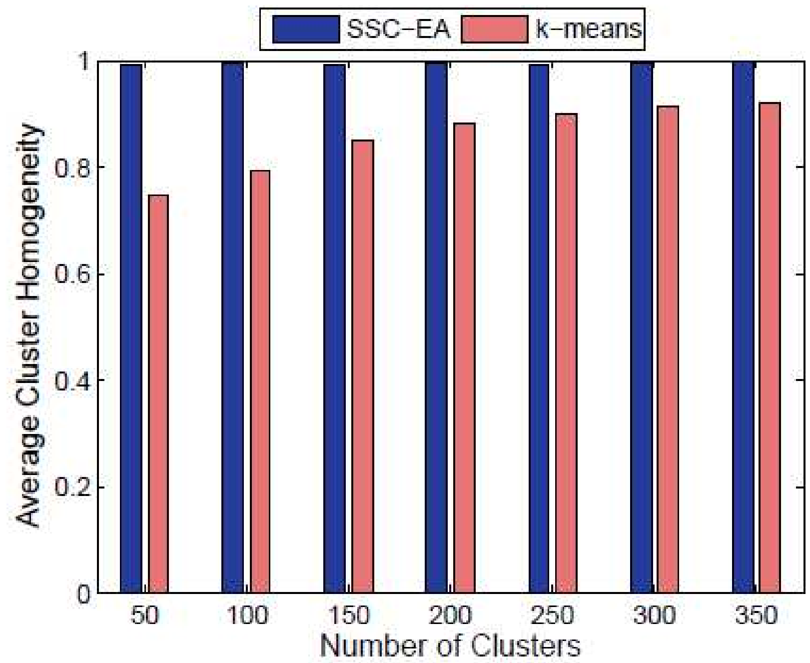
- $TP(k)$: correctly classified flows in cluster k ($\lambda = 100\%$).
- $n(k)$: number of flows in cluster k.
- n_{cls}: number of clusters.

- evaluations performed in UNIBIS.
- SSC-EA vs traditional clustering: DBSCAN and k-means.
- evaluate the impact of Feature Selection (FS) in clustering algorithms.

SSC-EA vs DBSCAN vs k-means



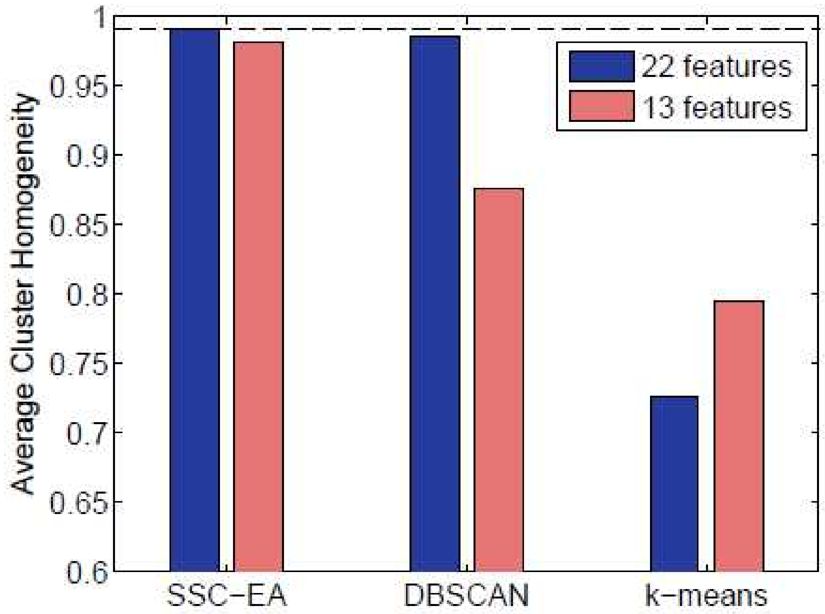
(a) GA: SSC-EA vs DBSCAN.



(b) ACH: SSC-EA vs k-means.

- SSC-EA is more robust than DBSCAN regarding clusters' size.
- SSC-EA achieves almost perfect ACH, highly improving k-means.
- SSC-EA GA is about 85%, with about 50 identified clusters.
- SSC-EA GA is impacted by some big-clusters with poor homogeneity.

Impacts of Feature Selection (FS) - Masking Features.



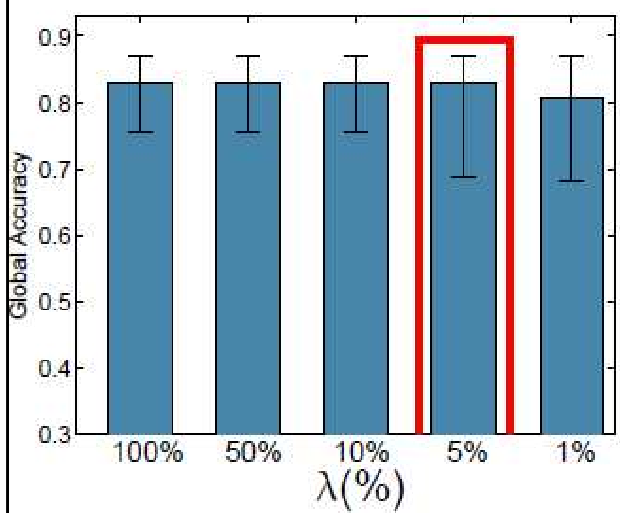
(a) Average per-cluster homogeneity.

(b) Global accuracy.

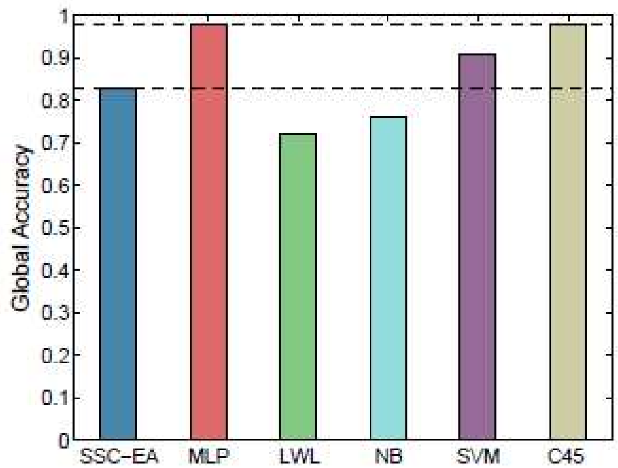
- GA for the 22 features, and a reduced set of 13 features obtained by FS.
- Selected features correspond mainly to flow volume and packet size features (independent of network conditions).
- SSC-EA is more robust against irrelevant or redundant features.
- The number of SSC-EA clusters falls to about 30 with 13 features.

Semi-Supervised vs Supervised Classification

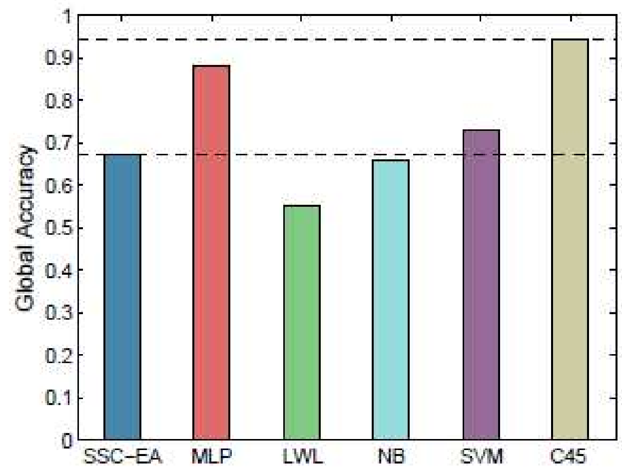
- The GA of SSC-EA slightly varies with λ (high homogeneity).
- Compare SSC-EA ($\lambda = 5\%$) against "full" supervised classifiers ($\lambda = 100\%$): C45, SVM, Neural Networks (NN), Bayes, and LWL.



(a) GA(λ)



(b) GA (UNIBIS)



(c) GA (VALTC)

- Difficult to compete with C45, SVM, NN (full training set, $\lambda = 100\%$).
- But limited labeled traffic provides a means for operational deployment.
- Periodically run SSC-EA to recalibrate the limited-reference classifier.

Automatic Traffic Classification

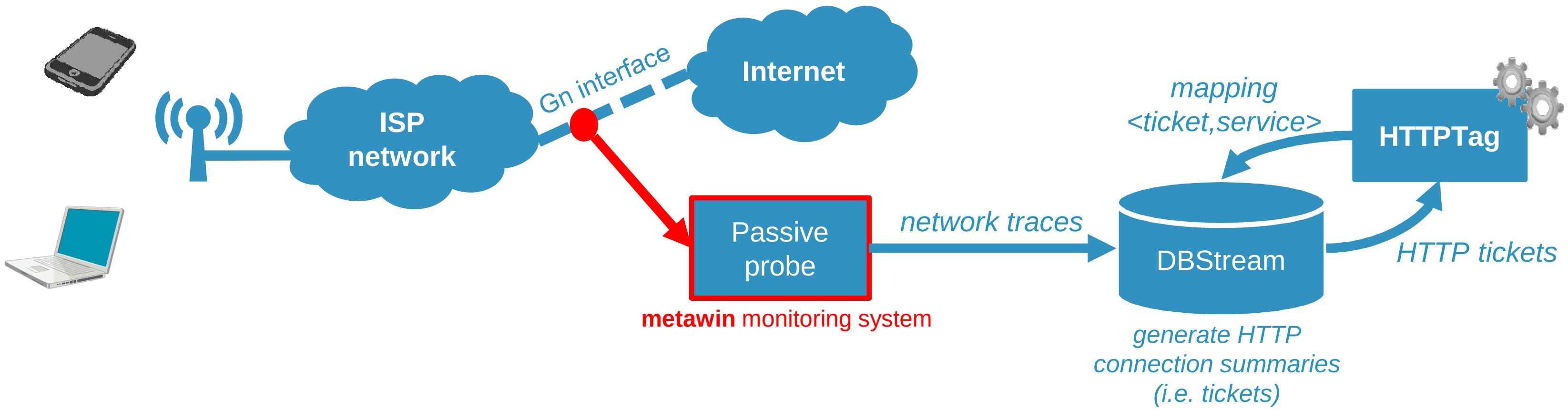
How to Detect Apps running on top of HTTP?

Classifying HTTP and HTTPS Apps

- So far we analyzed generic applications with their own protocols
- And what about apps embedded on HTTP?
  - How to get Facebook, Twitter, WhatsApp
  - Over HTTPS?
  - When served by the same CDN?
- And for the application?
  - Is it a video seen on Facebook?

HTTP Classification with HTTPTag [1/3]
Introduction

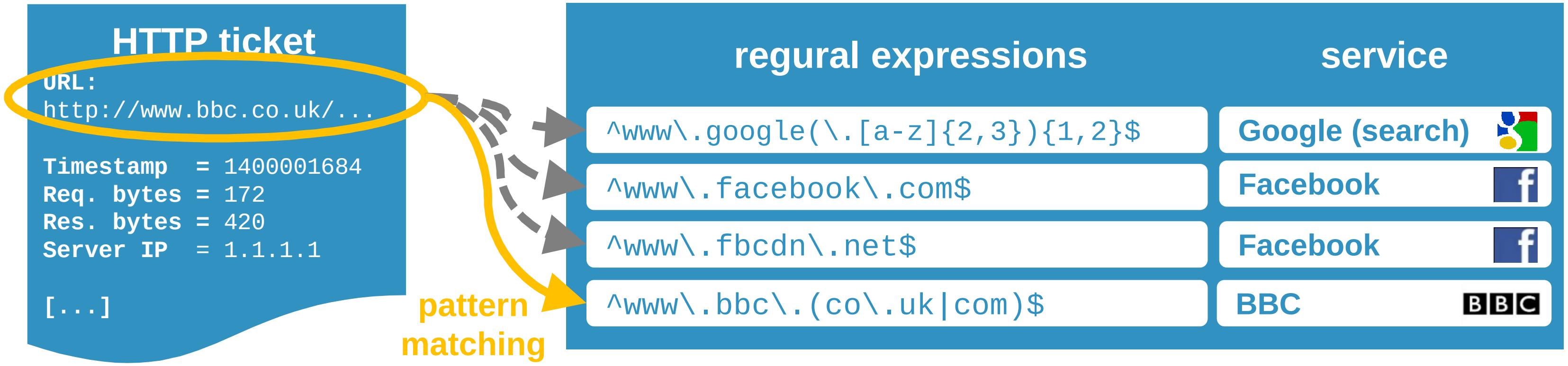
- First step in network analysis: service classification



- On-line classification system for HTTP-based traffic, running on top of METAWIN 3G/4G monitoring system
- Reads only HTTP headers (no DPI)
- Labels HTTP flows by analysing the conttacted hostnames

HTTP Classification with HTTPTag [2/4]

Pattern matching



Example: Facebook regex

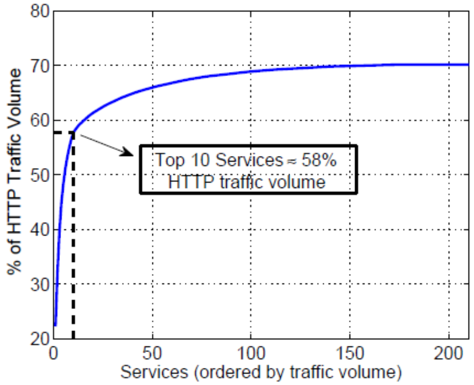
`(((|%.)(facebook.com|fbcdn.net))|((fbcdn|fbstatic)%.akamaihd.net))`

- Manually defined patterns (initial effort, but high stability)
- Flows classified by **pattern matching** on the requested URL
- Easy to discover new popular web services

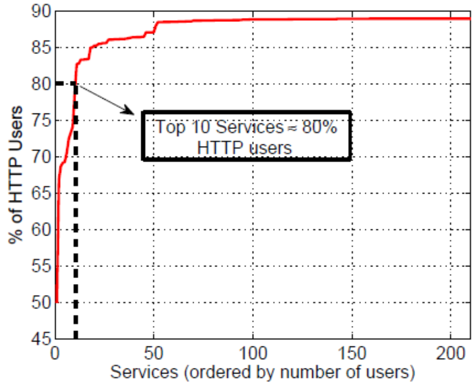
- HTTPTag allows to associate **server IPs** to the recognized web service
  service $S \rightarrow A = \{S, IP\}$

HTTP Classification with HTTPTag [3/4]

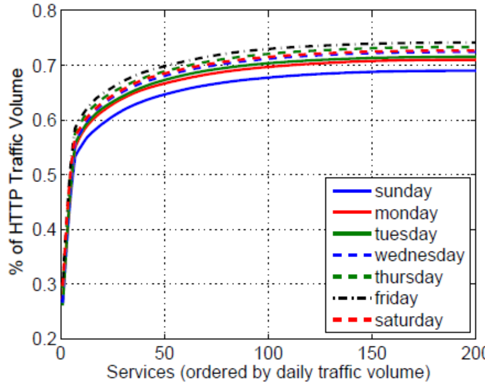
Classification Rate



HTTP traffic volume per service



Unique HTTP users per service



Daily traffic volume per service

- Using **280 labels** (i.e. services), **HTTPTag classifies 70%** of the **HTTP traffic volume accessed by 88% of the customers** in an operational 3G network

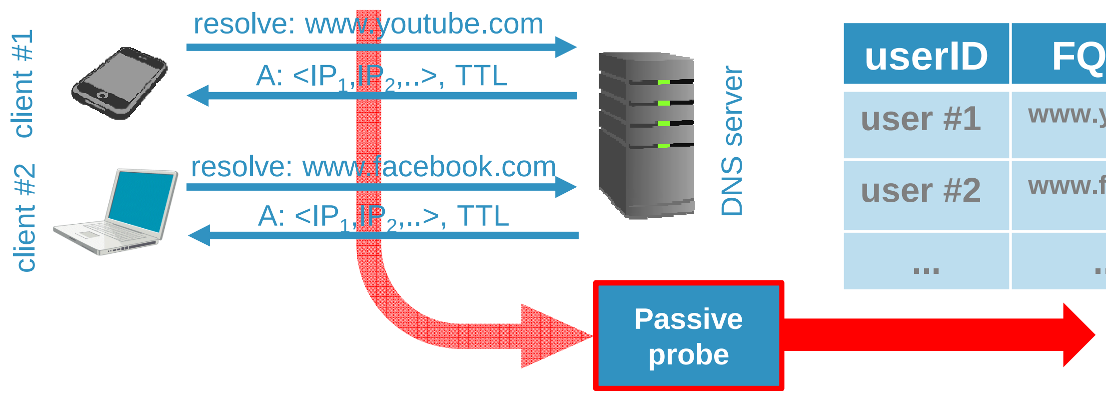
- **Elephant services:** the **top-10 services** account for almost **60% of HTTP traffic volume**, and are **accessed by 80% of the customers**

- Top services: **YouTube, Facebook, Google Search, Apple (iTunes Store and AppStore), Adult Video Services, Windows Update Services**, etc.

HTTP Classification with HTTPTag [4/4]

Leveraging DNS for HTTPS classification

- HTTP header pattern matching inapplicable for **HTTPS** (encrypted!)
- Idea: use **passively collected DNS** requests to dynamically map *<services,serverIPs>*



local mapping

validity period (TTL)

userID	FQDN	IP	start	end
user #1	www.yt.com	1.1.1.1	14041347	14041349
user #2	www.fb.com	2.2.2.2	14030424	14031288
...	...	...	...	...

- Every subsequent flow between a **<user>** and a **<server_ip>** in the validity period **[validity_start:validity_end]** are assigned to **<FQDN>**
- The Fully Qualified Domain Names (FQDN) are assigned to service with usual pattern matching

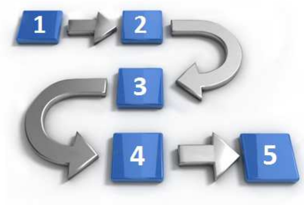
Automatic Traffic Classification

Mini – IPC: Classifying HTTP flows from IP addresses

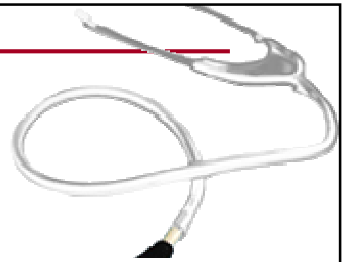
HTTP Classification with IP – FQDN mapping

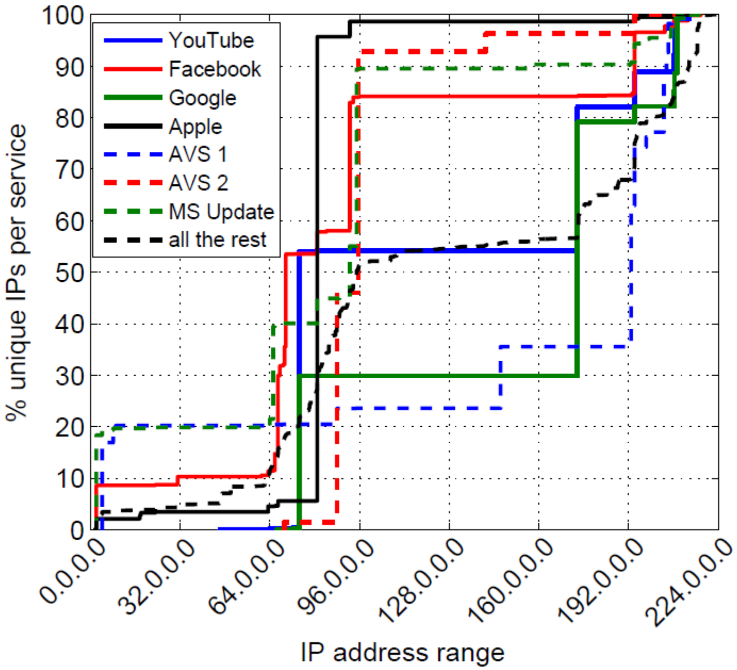
- In a nutshell, Mini-IPC classifies HTTP flows based **solely** on the IP address of the server being contacted.
- Given a specific service S_i to classify, Mini-IPC builds a set of k_i IP addresses $IP_i = \{ip_i(1), ip_i(2), \ldots, ip_i(k_i)\}$ hosting S_i…
- …using the associations $A_i = \{S_i, IP_i\}$ between server IPs and services provided by HTTPTag during a **learning phase**
- **Classification phase:** given a list of m services $S_i = \{i=1..m\}$ to classify and a new flow f_{new} from ip_{new} :

$$F(f_{new}) = S_i \leftrightarrow ip_{new} \in IP_i$$

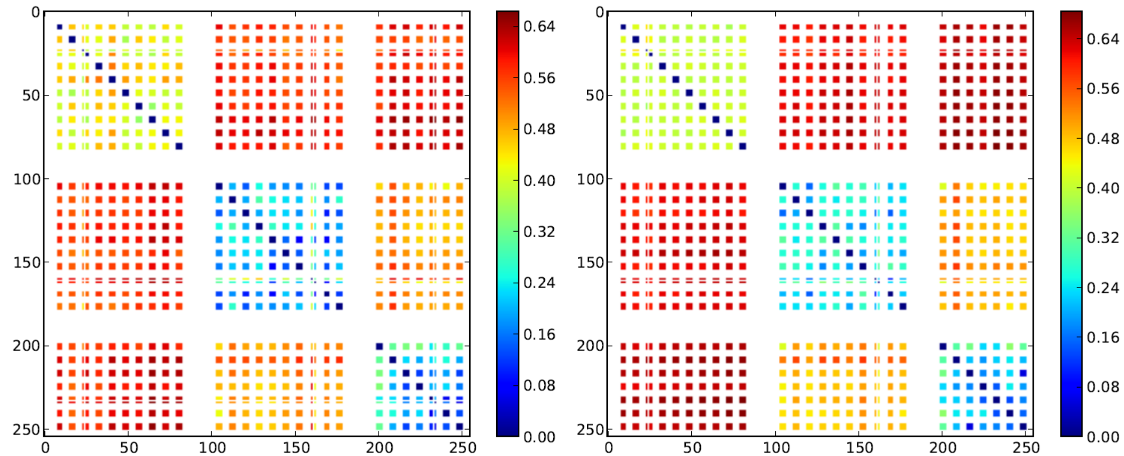


IP Collisions and Mappings‘ Stability





(a) IP range distribution on a single day



(a) 2 days (b) 10 days

IPs–services stability in Akamai, for Facebook

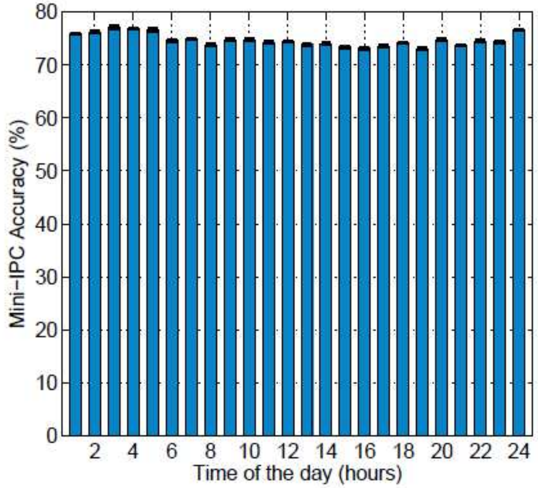
- **IP collisions** → different services are provisioned by the same IP address at different times of the day (same CDN, datacenter front-end, IP anycast, etc.)
- For example, **Google Search** and **Facebook** collide, as well as **Facebook with Apple Services and Windows Services**
- **Yet, some regions of the Akamai IP space are very stable and used exclusively for some services**

Evaluation of Mini-IPC

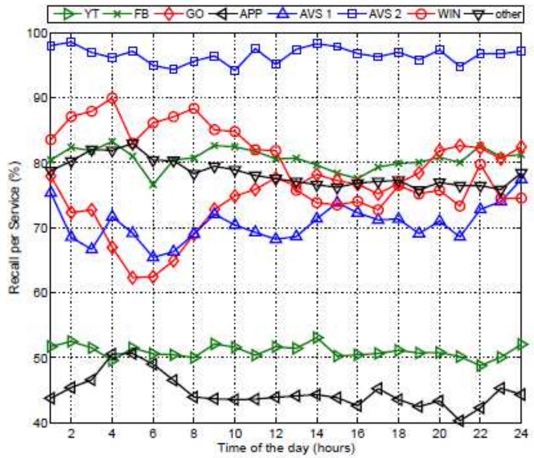
- 8-classes classification problem → **top-7 HTTP services**
- The rest of the labeled traffic belongs to the **Other class**
- If HTTP flow $\notin$ class i, i=$1..7$ → assign class Other
- **If IP collision** → random selection among the collided clases (20 runs to avoid random results)
- #IP = {1373, 2031, 1875, 522, 92, 456, 743} for top-7 services
- Classification Accuracy (CA)
- Recall & Precision (per class)

$$\mathrm{CA} = \frac{\sum_{i=1}^{m} TP_i}{n}, \quad \mathrm{R}_i = \frac{TP_i}{TP_i + FN_i}, \quad \mathrm{P}_i = \frac{TP_i}{TP_i + FP_i}$$

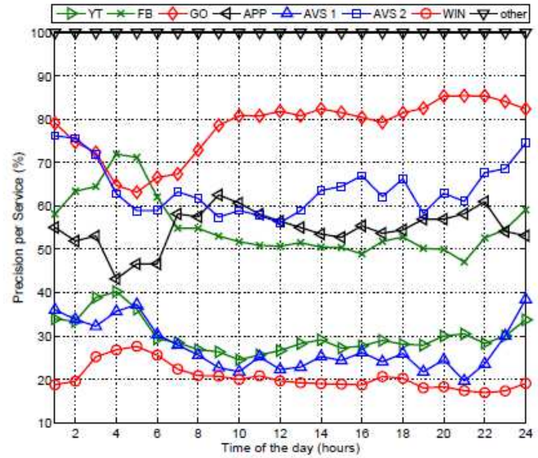
Evaluation of Mini-IPC – 1 day learning/testing



(a) Mini-IPC Accuracy.



(b) Mini-IPC Recall per Service.

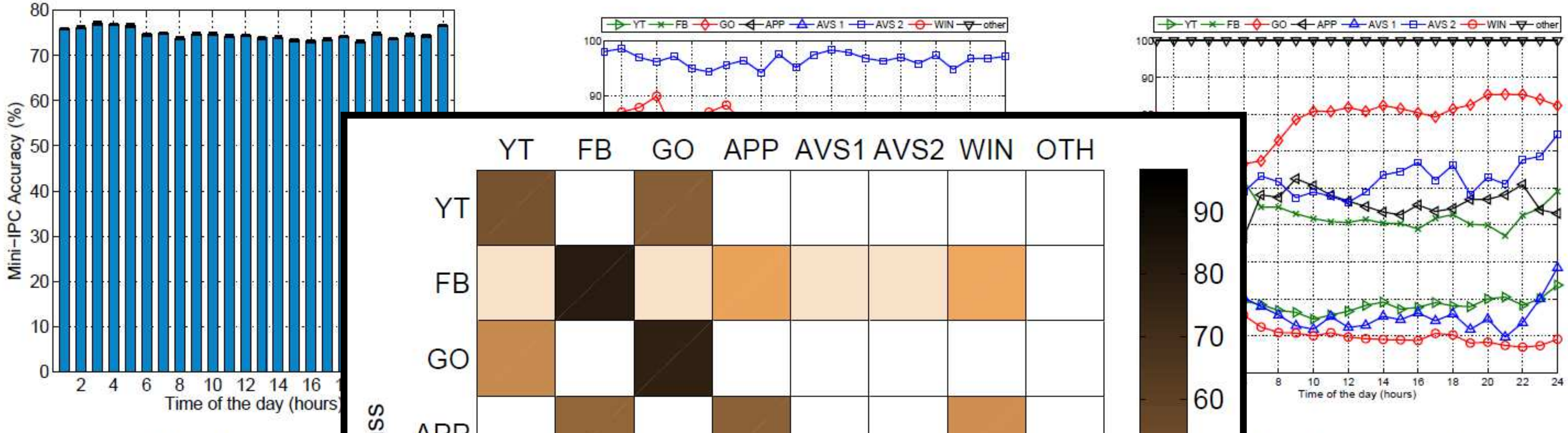


(c) Mini-IPC Precision per Service.

- The **classification accuracy** is high and **stable during the day**, close to **75% of correctly classified HTTP flows**
- More than **60%** of all the **Facebook, Adult Video, Google Search,** and **Win Update** HTTP flows are **correctly classified**
- Precision for Google flows is still pretty high and above 80% from 9 am onwards, but results for YouTube, AVS 1, and Win Update show a **big number of false positives (IP collisions)**

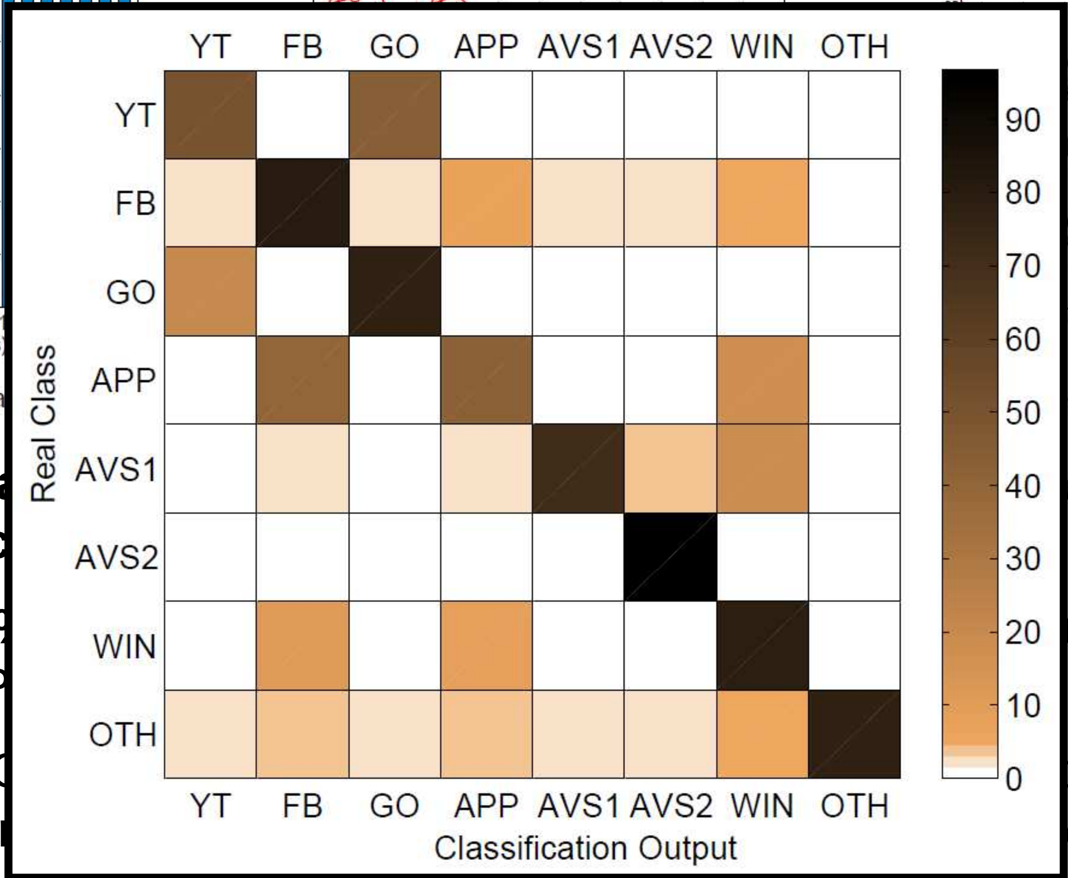
***Note: recall & precision are unbalanced by the Other class**

Evaluation of Mini-IPC – 1 day learning/testing



(a) Mini-IPC Accura

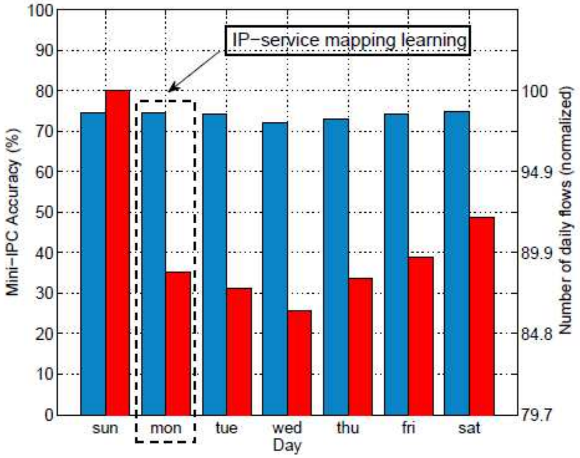
-IPC Precision per Service.



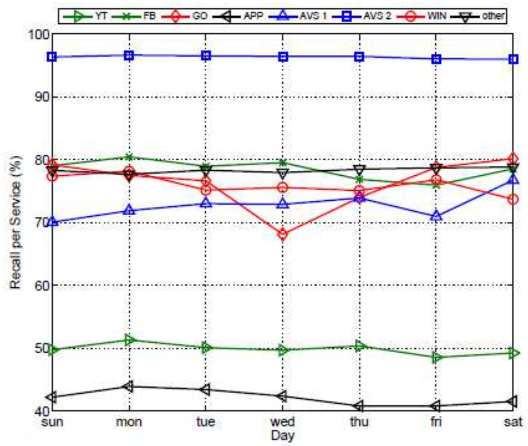
- The **classifica** ... **y**, close to **75%** **of correctly c**
- More than **60%** ... **rch,** and **Win Update** HTTP
- Precision for ... om 9 am onwards, but ... a **big number of false positives (IP collisions)**

***Note: recall & precision are unbalanced by the Other class**

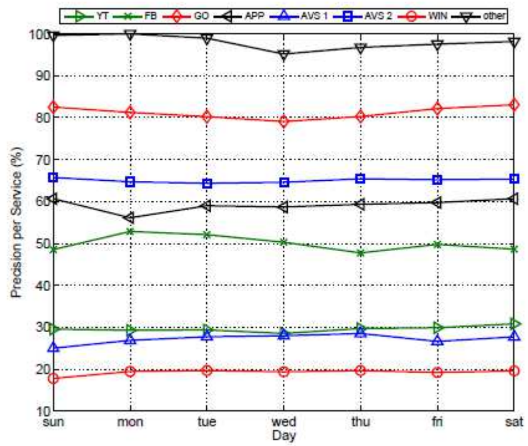
Evaluation of Mini-IPC – 1 day/1 week learning/testing



(a) Mini-IPC Daily Accuracy and num Flows.



(b) Mini-IPC Daily Recall per Service.



(c) Mini-IPC Daily Precision per Service.

- **Results remain almost unchanged for the evaluation on the full week,** even if strong variations might be observed in the # HTTP flows (e.g. Sun)

- This may suggest that the **sets of IPs** provisioning the different services are **stable in time**, **at least in a weekly-basis**

DNS to the rescue

Classifying HTTPS traffic through DNS analysis

Use case scenario – The “boss” view

- The boss asks to netadmin to
  - allow Facebook but to block Zynga gaming platform
  - YouTube (as aggregate) should not exceed 10Mbps
  - improve Gmail and Dropbox performance

- …but nowdays services are complex
  - Encryption++
  - CDN++
  - Cloud++

No DPI
No IP servers info
Time-variant policies

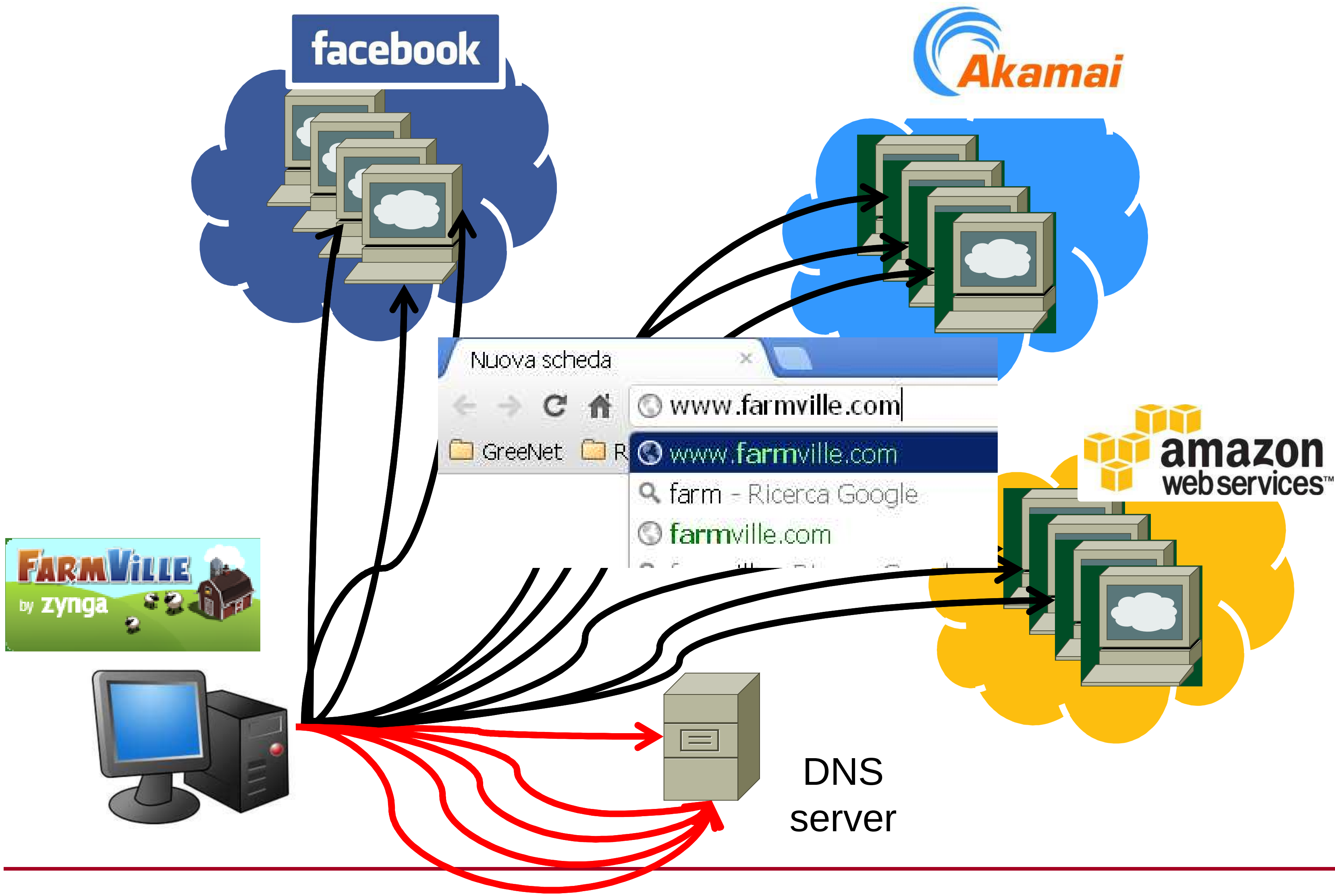
Use case scenario – The "netadmin" view

- netadmin sees lot of requests going to 73.194.78.141
  - `wg-in-f141.1e100.net` → owned by Google
  - Protocol is unknown (binary, maybe encrypted?)
- Should netadmin block it?
  - What if it is related to `www.google.com` ?!?!
- FarmVille by zynga lives on Facebook and runs on amazon web services, Akamai
  - …but also Dropbox uses amazon web services
  - netadmin's firewall would either block both or let everyone enjoy Farmville!

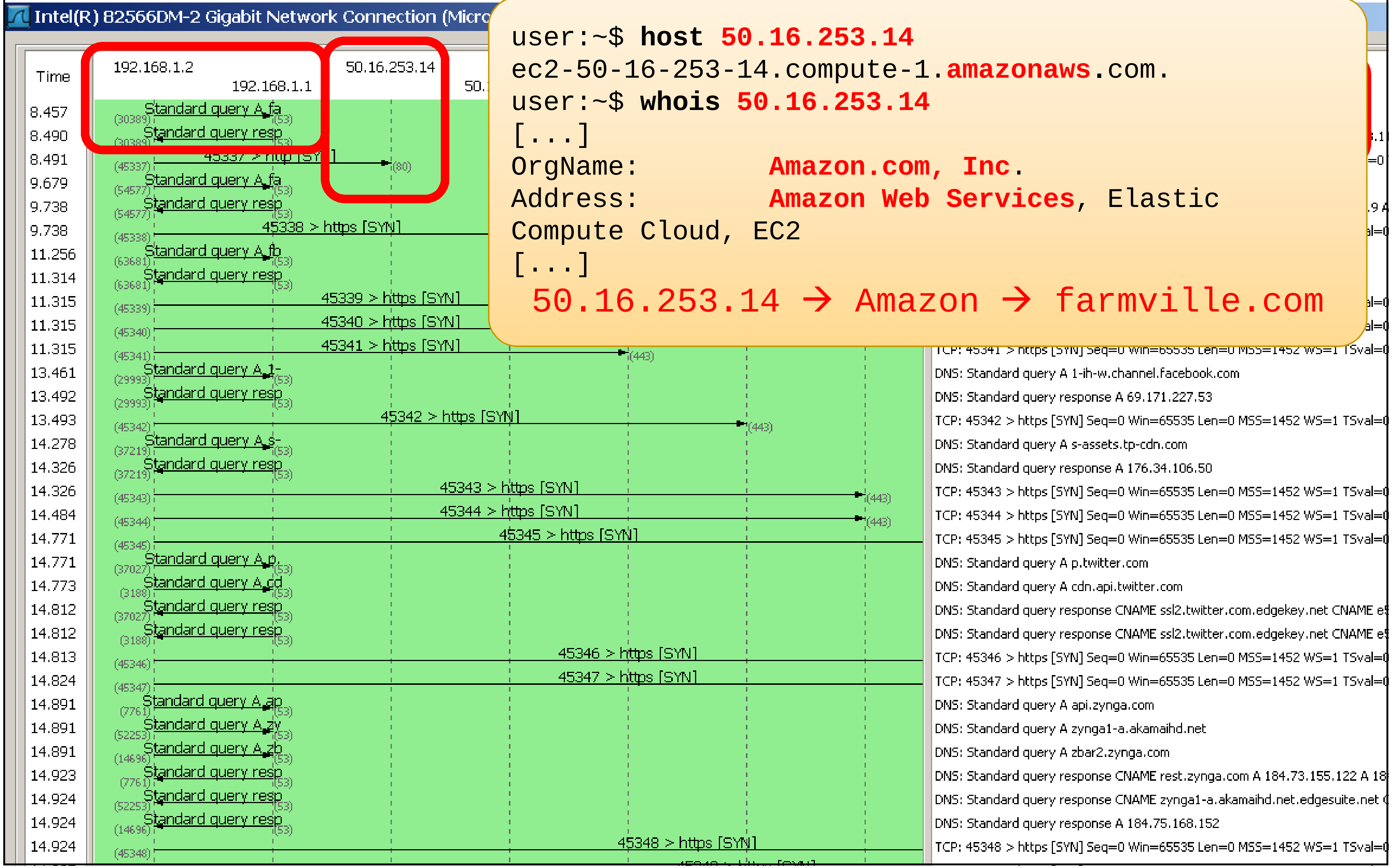
But wait a second…

DNS messages carry the mapping between logical names and IP addresses…

The intuition



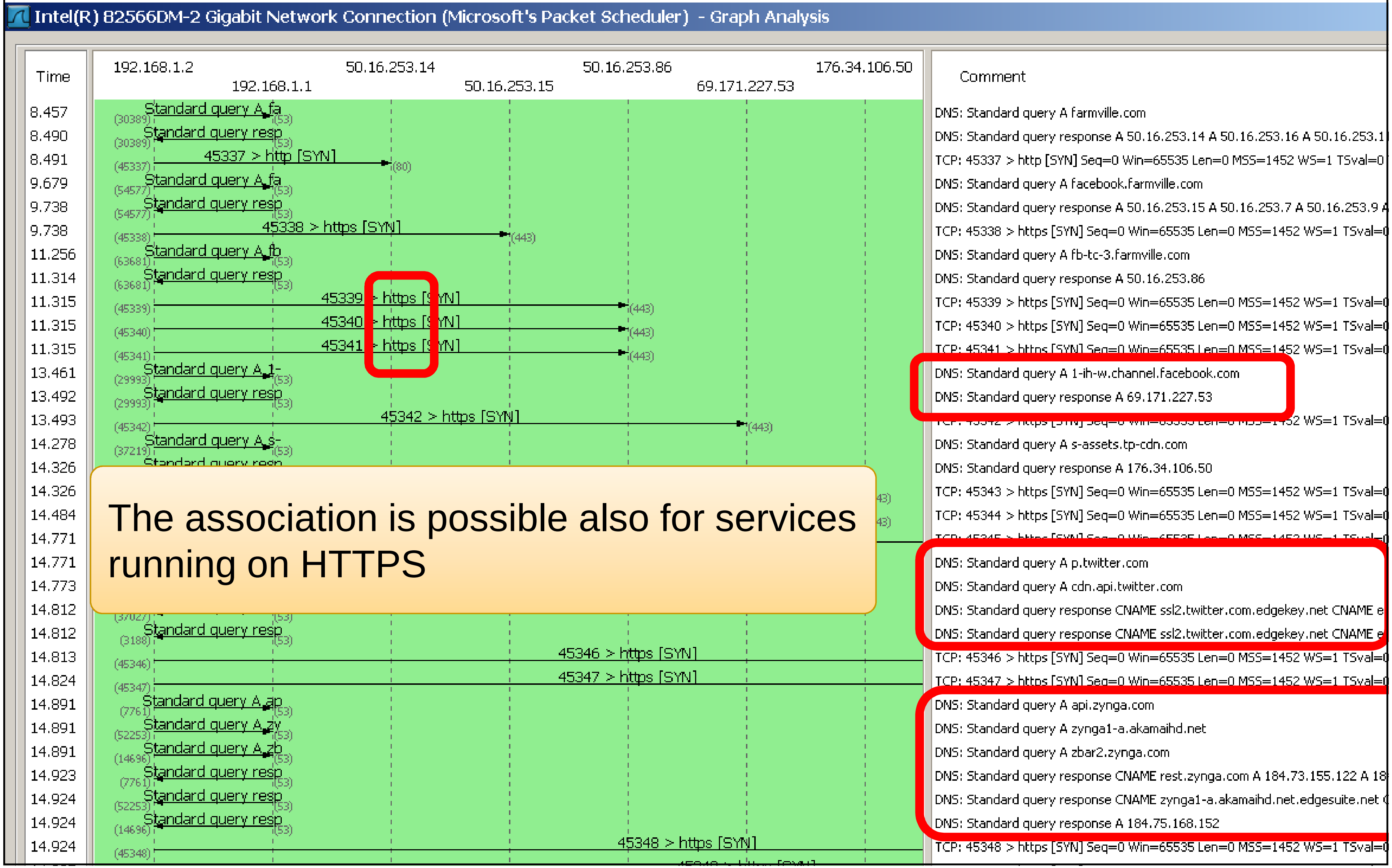
The intuition




```
user:~$ host 50.16.253.14
ec2-50-16-253-14.compute-1.amazonaws.com.
user:~$ whois 50.16.253.14
[...]
OrgName:        Amazon.com, Inc.
Address:        Amazon Web Services, Elastic
Compute Cloud, EC2
[...]
```

50.16.253.14 → Amazon → farmville.com

The intuition

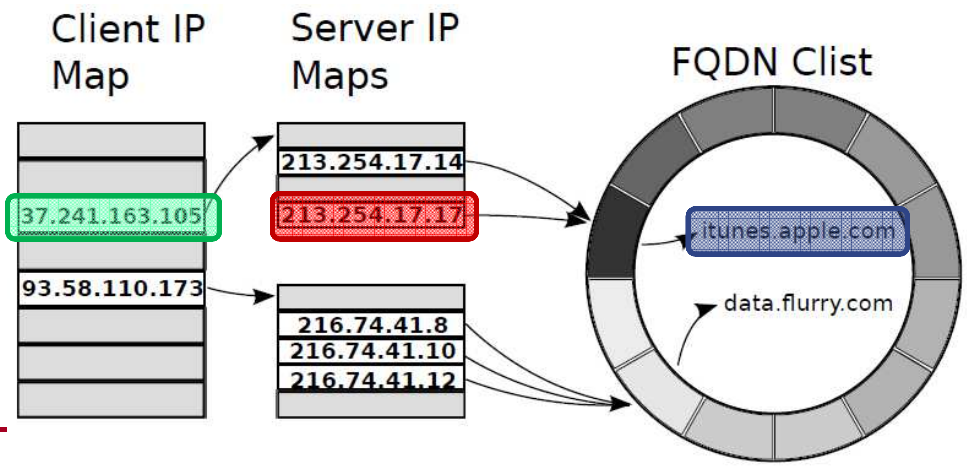


The association is possible also for services running on HTTPS

DNS to the rescue

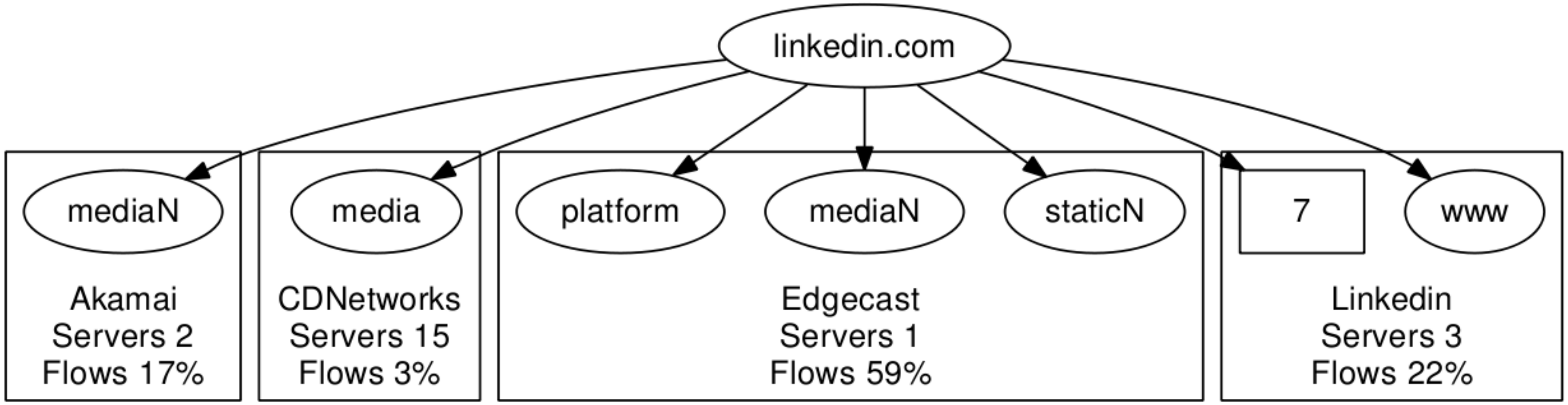
- Correlating flows IPs with DNS queries will provide a natural way of mapping content and traffic
  - Registered names usually carry some semantic
  - Many web/client-server applications use DNS to get the IP address of the target host
- For simplicity, it is implemented with
  - single buffer to store FQDN (no need to handle TTL)
  - access based on client and server IP

Tstat find a new flow
From **37.241.163.105**
To **213.254.17.17**



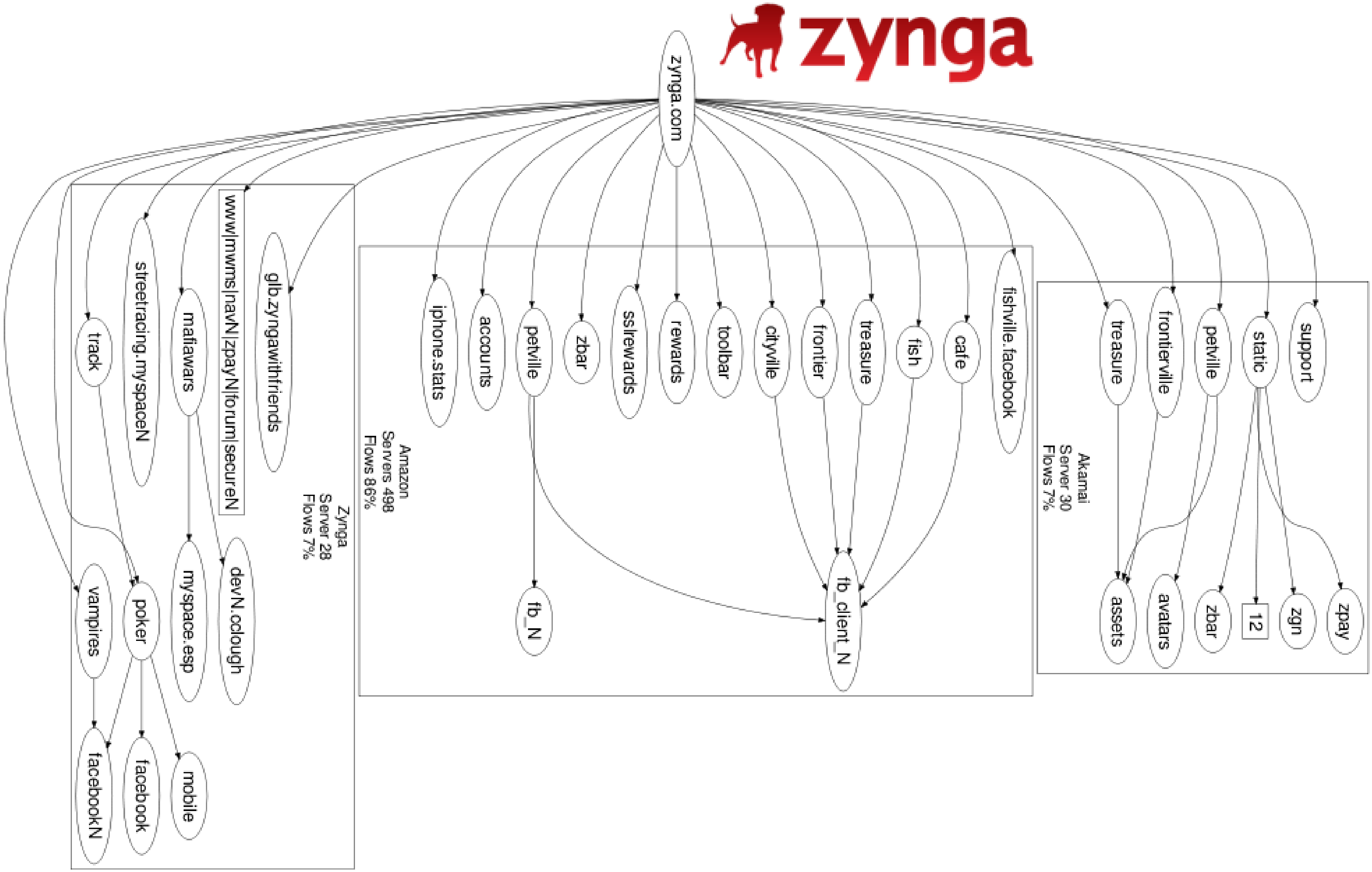
Example: services (de)composition (1/3)

- How LinkedIn domain is composed?



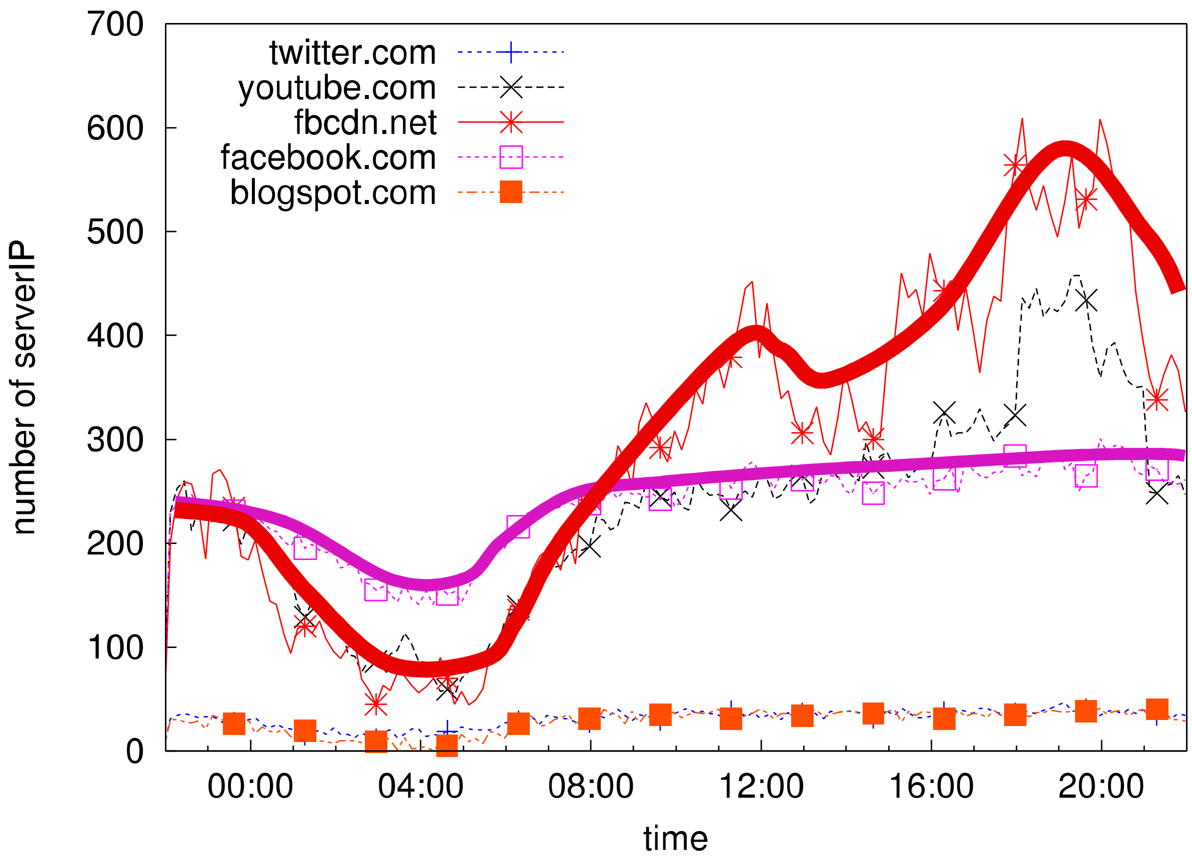
…but this is simple example

Example: services (de)composition (2/3)



Example: time-variant patterns (3/3)

- Evolution of #IP used over the day

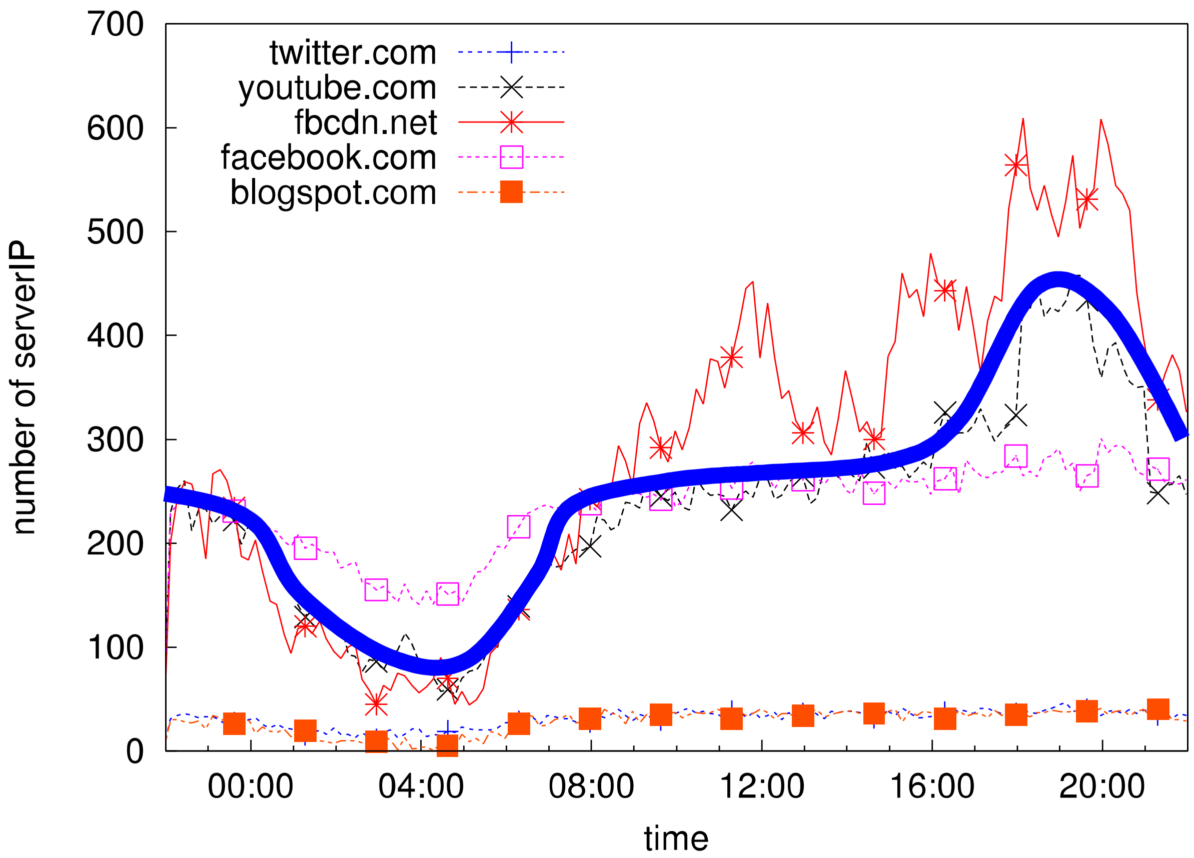


fbcdn.net = Facebook-Akamai
Is highly variable

facebook.com = Facebook Inc.
needs only 300 IPs

Example: time-variant patterns (3/3)

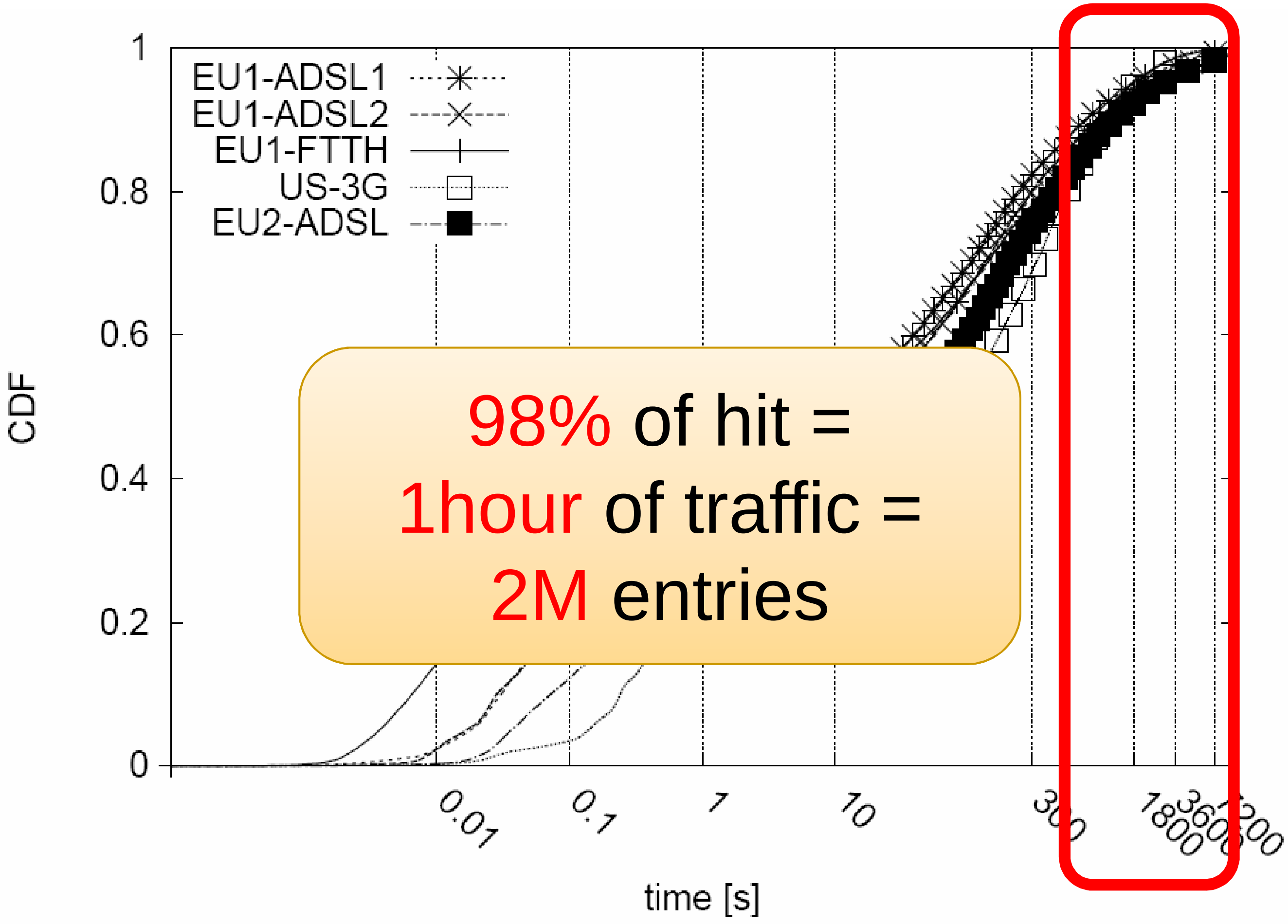
- Evolution of #IP used over the day



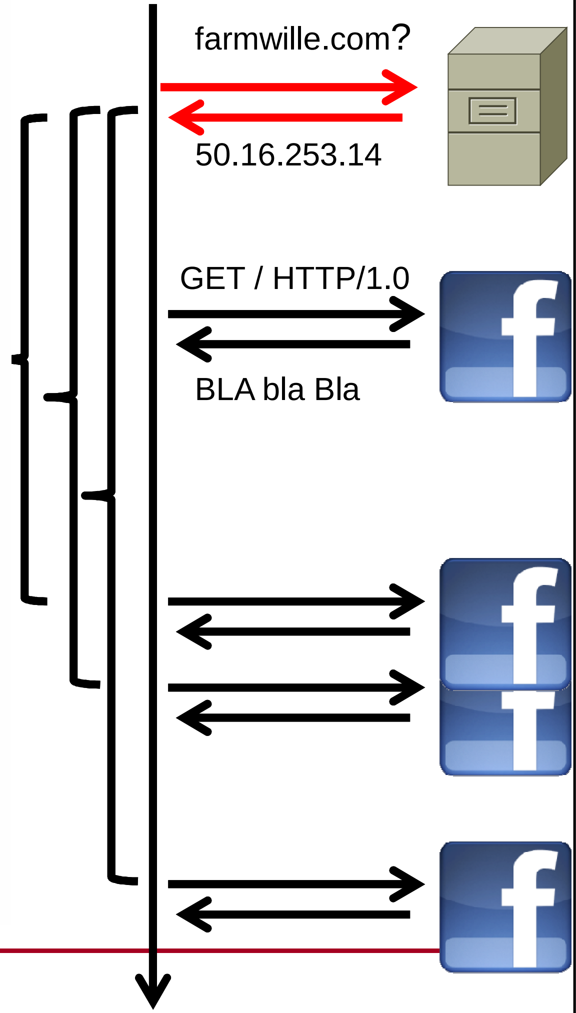
youtube.com presents a shift in the evening (load?)

How big should be the cache?

After the resolution, for how long the mapping is useful?



98% of hit =
1hour of traffic =
2M entries



Reverse engineering Whatsapp naming scheme
Hybrid measurements

ftw Creating Communication Technologies

Testbed:

- Traffic (chat and medie exchange) actively generated at end devices (Android and iOS)
- Passively captured at a gateway (**Wireshark**)
- Focus on DNS requests

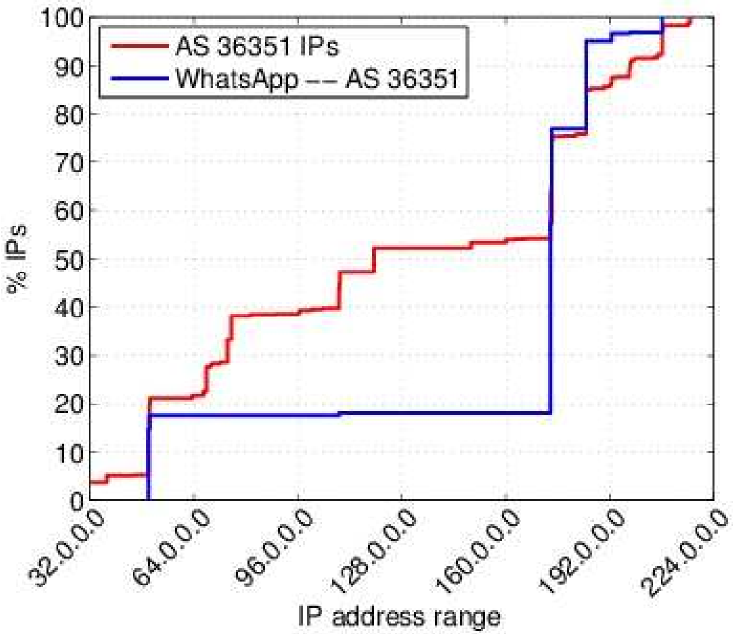
Findings:

- Whatsapp used custom XMPP protocol
- Media exchange via HTTPS servers
- One persistent SSL connection to XMPP servers while the app is running
- Dedicated TLS connections to HTTPS servers for each media transfer

Servers naming scheme:

domain	prot. (port)	type
`cX, eX, dX`	XMPP(5222,443)	chat & control
`mmiXYZ,mmsXYZ`	HTTPS (443)	media (photo,audio)
`mmvXYZ`	HTTPS (443)	media (video)

Revealing Hosting Infrastructure
Through large-scale passive measurements



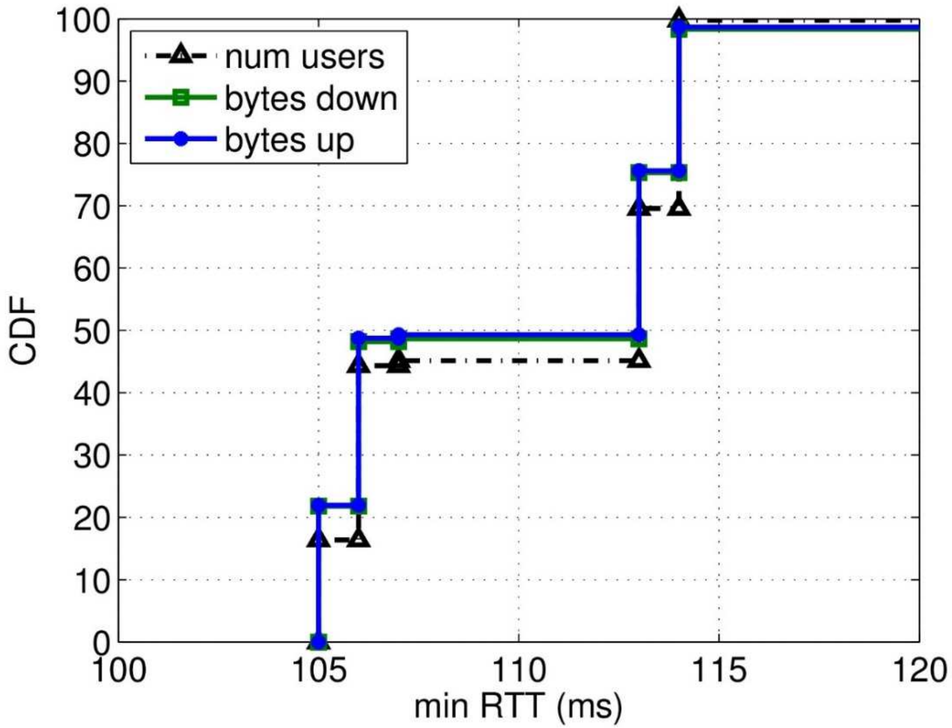
- 386 IP adresses used by Whatsapp (chat and media)
- All in AS36351 (Softlayer)

SOFTLAYER® an IBM Company

Service/AS	#IPs	# /24	# /16	# /8
WhatsApp	386	51	30	24
SoftLayer (AS36351)	1364480	5330	106	42

ftw Creating Communication Technologies

Localization of servers through RTT measurements

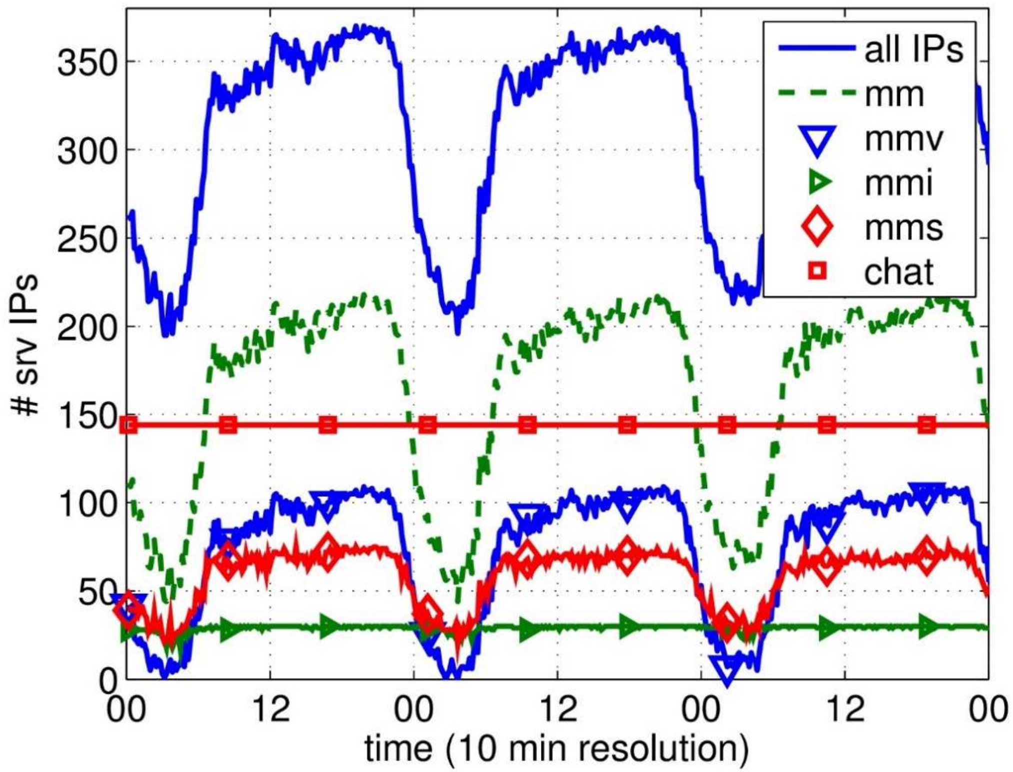


- ~400 IP addresses in Softlayer AS
- Two big steps in RTT distribution at 106ms and 114ms
- Localized by MaxMind in **Houston** and **Dallas** (Texas)

MAXMIND

Revealing Hosting Infrastructure
Through large-scale passive measurements

Active IPs



- More than 350 IPs during peak hours
- At least 200 IPs always active (chat servers)
- ~25 IPs always active (`mmi` servers)

RIPE Atlas infrastructure for geo-distributed active measurements

- **RIPE NCC**: Regional Internet Registry for Europe

- **RIPE Atlas**: a large measurement network composed of geographically distributed active probe used to measure connectability and reachabiltiy

http://atlas.ripe.net

RIPE Atlas probe v3
TP-Link MR3020 router with custom firmware

Hosting infrastructure
Geographical distributed active measurements

http://atlas.ripe.net

RIPE Atlas probe v3
TP-Link MR3020 router with custom firmware

- My UDM (User Defined Measurement): 600 probes world-wide resolve Whatsapp hostnames (`{mmX|dX}.whatsapp.net`)
- Result: same set of IP addresses

Previous conclusions for WhatsApp hosting infrastructure are still valid from other VPs