Predicting Real-time Service-level Metrics from Device Statistics

Rerngvit Yanggratoke[1]

In collaboration with
Jawwad Ahmed[2], John Ardelius[3], Christofer Flinta[2], Andreas Johnsson[2], Zuoying Jiang[1], Daniel Gillblad[3], Rolf Stadler[1]

(1) KTH Royal Institute of Technology, Sweden
(2) Ericsson Research, Sweden
(3) Swedish Institute of Computer Science (SICS), Sweden

Machine Learning meetup at Spotify

Stockholm, Oct 21, 2015

About Me

- Currently, PhD student at KTH
- “Real-time analytics for network management”
  - Apply machine learning on system-generated data to better manage networked services and systems
- Favorite data science tools
  - R for data analysis and visualization
  - SparkR for large-scale batch processing
- Free time
  - Playing foosball and badminton
  - Developing and playing mobile games
  - Contributing to Opensource projects: Flink and SparkR

Overview

- Video frame rate, web response time, read latency, map/reduce job completion time, ….

Services

….

Operating System

Hardware

Real-time service metrics Y

Predict

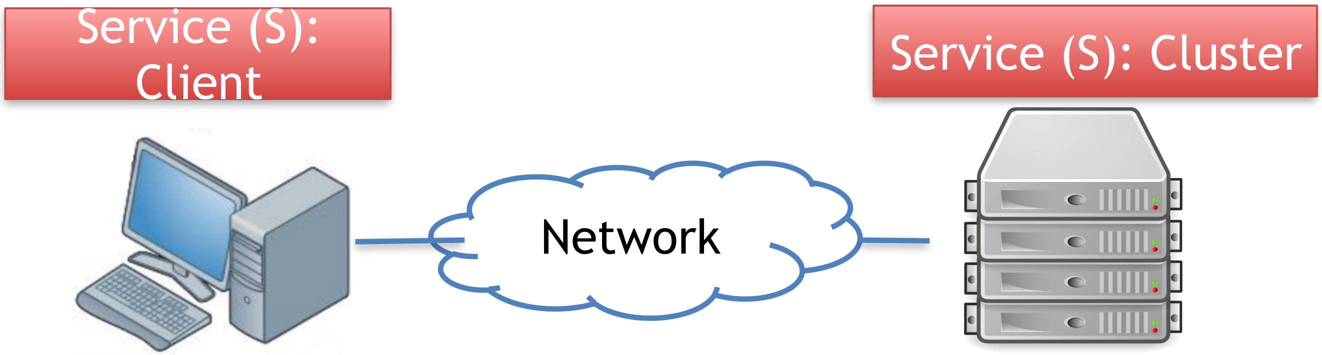
Device statistics X

- CPU load, memory load, #netwo active sockets, #context switching, #processes, etc..
- We read raw data from /proc provided by Linux kernel

Outline

- Problem / motivation
- Device statistics X
- Service-level metrics Y
- Testbed setup
- X-Y traces
- Data cleaning and preprocessing
- Evaluation results
- Real-time analytics engine
- Demo
- Recap

Problem / motivation



Y: service-level metrics

- Video frame rate, audio buffer rate, RTP packet rate
- We select Video streaming (VLC) as an example service

X: device statistics

- CPU load, memory load, #network active sockets, #context switching, #processes, etc..

Problem : **M: $X \rightarrow \hat{Y}$** predicts Y in real-time

[Supervised-learning regression problem]

Motivation :

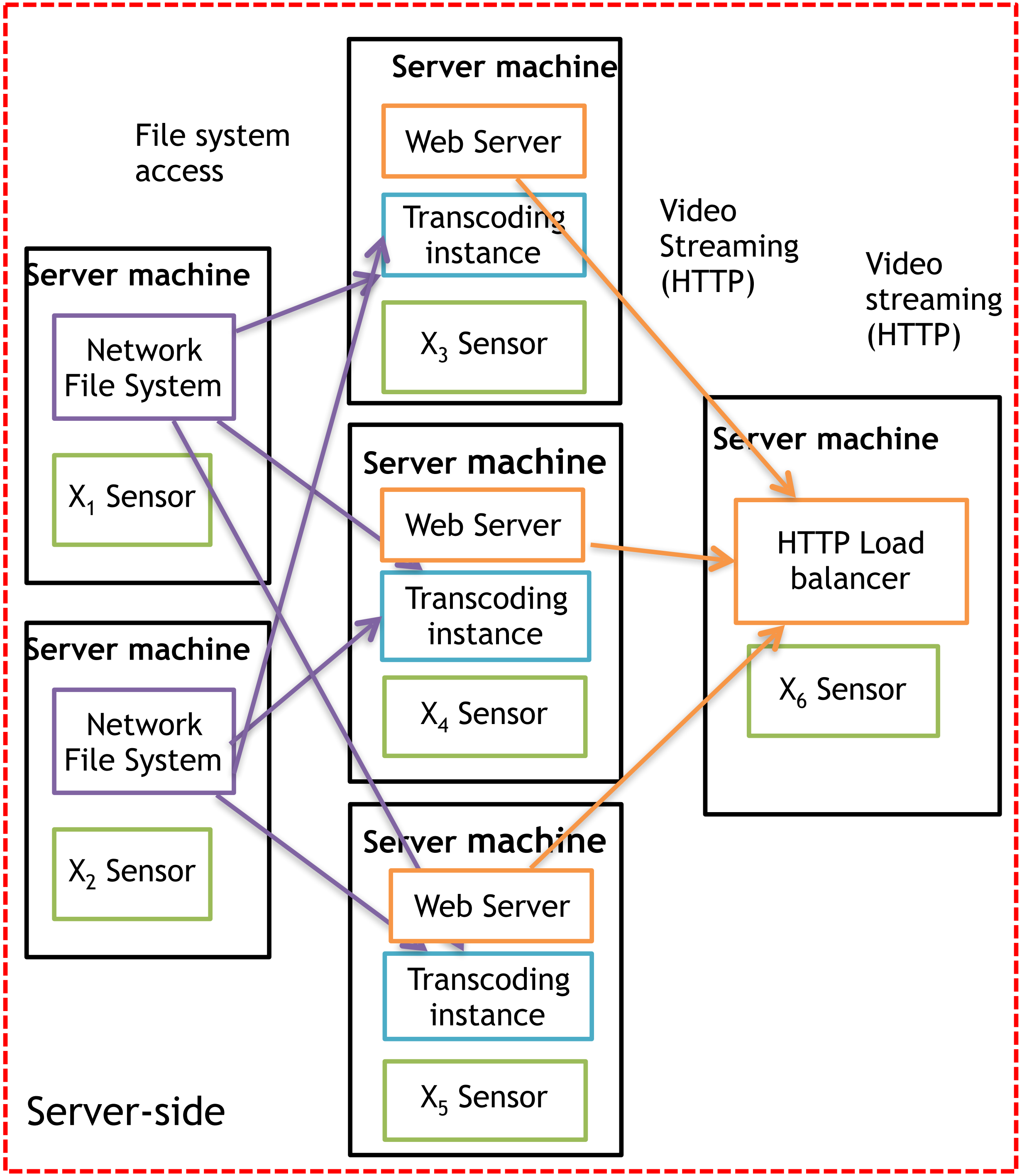
- Building block for real-time service assurance for a service operator or infrastructure provider

Device statistics X_{proc} and X_{sar}

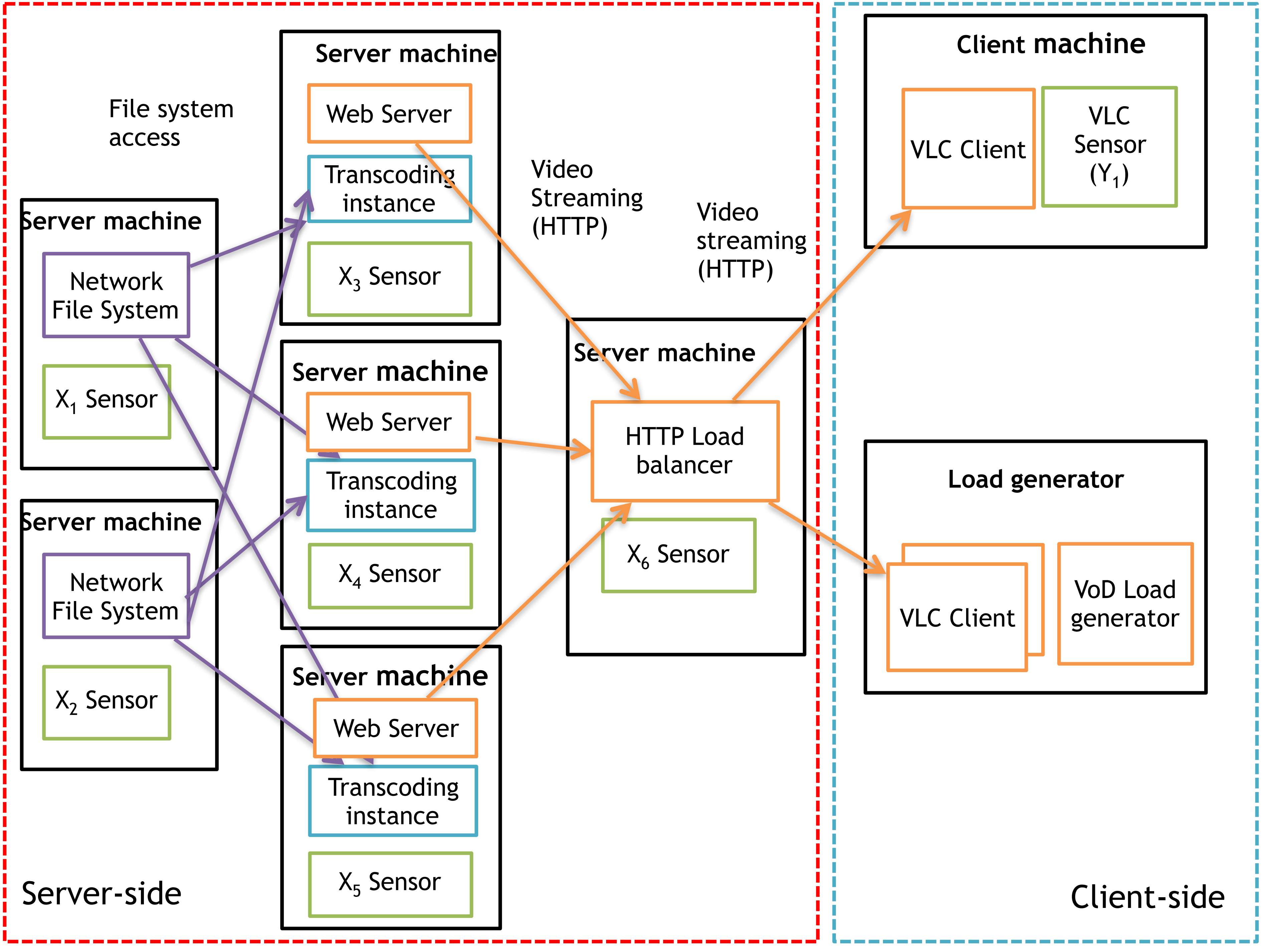
- Linux kernel statistics X_{proc}
  - Features extracted from /proc directory
  - CPU core jiffies, current memory usage, virtual memory statistics, #processes, #blocked processes, …
  - About 4,000 metrics
- System Activity Report (SAR) X_{sar}
  - SAR computes metrics from /proc over time interval
  - CPU core utilization, memory and swap space utilization, disk I/O statistics, …
  - About 840 metrics
- X_{proc} contains many OS counters, while X_{sar} does not
- For model predictions, include numerical features

Service-level metrics Y

- Video streaming service based on VLC media player
- Measured metrics
  - Video frame rate (frames/sec)
  - Audio buffer rate (buffers/sec)
  - RTP packet rate (packets/sec)
  - …
- We instrumented the VLC software to capture underlying events to compute the metrics.



File system access
Server machine
Network File System
X_1 Sensor
Server machine
Network File System
X_2 Sensor
Server machine
Web Server
Transcoding instance
X_3 Sensor
Server machine
Web Server
Transcoding instance
X_4 Sensor
Server machine
Web Server
Transcoding instance
X_5 Sensor
Video Streaming (HTTP)
Video streaming (HTTP)
Server machine
HTTP Load balancer
X_6 Sensor
Server-side

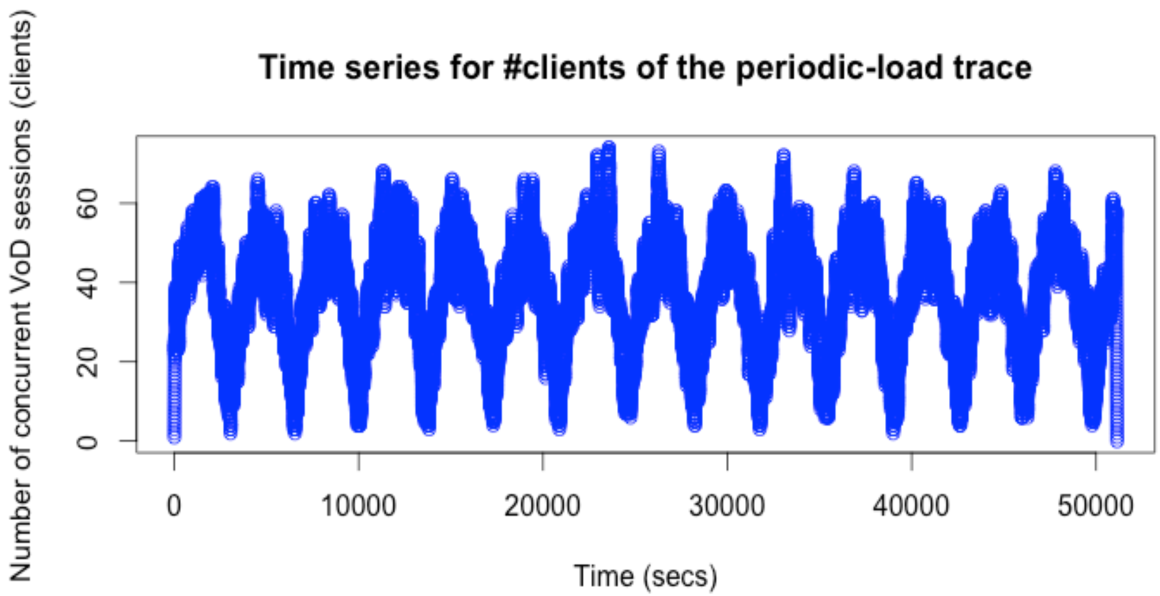


File system access
Server machine
Network File System
X_1 Sensor
Server machine
Network File System
X_2 Sensor
Server machine
Web Server
Transcoding instance
X_3 Sensor
Server machine
Web Server
Transcoding instance
X_4 Sensor
Server machine
Web Server
Transcoding instance
X_5 Sensor
Video Streaming (HTTP)
Video streaming (HTTP)
Server machine
HTTP Load balancer
X_6 Sensor
Client machine
VLC Client
VLC Sensor (Y_1)
Load generator
VLC Client
VoD Load generator
Server-side
Client-side

X-Y traces for evaluation

We collect the following traces

– Periodic-load trace, flashcrowd-load trace, constant-load trace, poisson-load trace, linearly-increasing-load trace



We published the traces use in our works

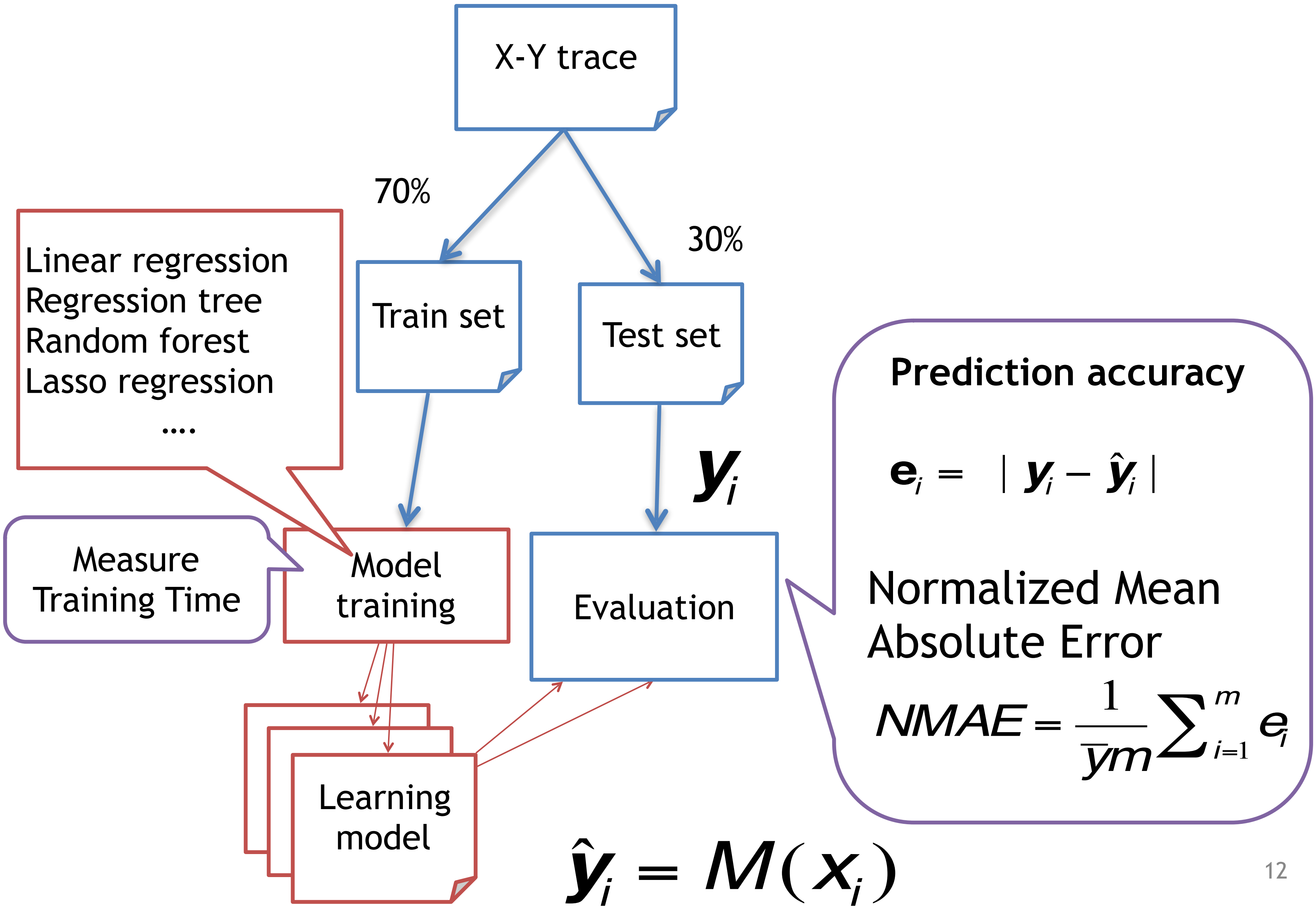
http://mldata.org/repository/data/viewslug/realm-im2015-vod-traces/

http://mldata.org/repository/data/viewslug/realm-cnsm2015-vod-traces/

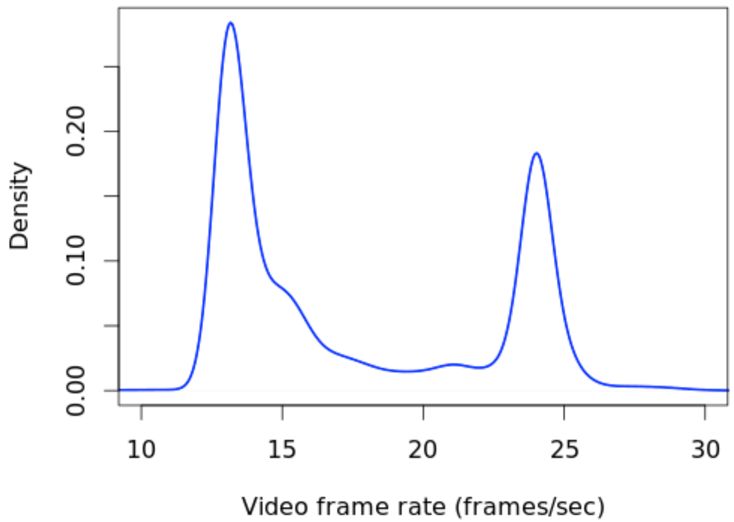
Data cleaning and preprocessing

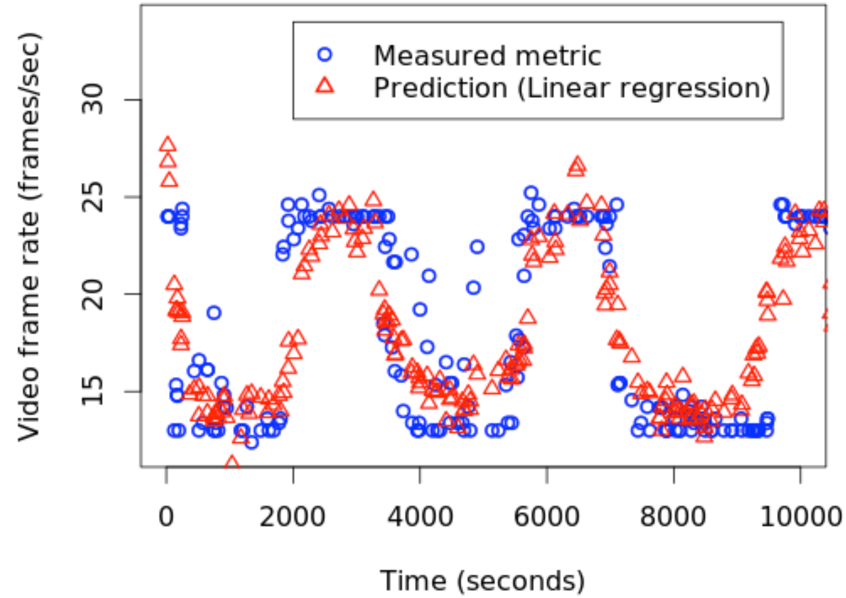
- Cleaning
  - Removal of non-numeric features
  - Removal of highly correlated features
  - Removal of constant features
  - …
- Preprocessing options
  - Principal component analysis
    - Dimensionality reduction
  - Incorporate historical data ($X = [X_t, X_{t-1}, \ldots, X_{t-k}]$)
  - *Automatic feature selection
  - ….

Model training and evaluation

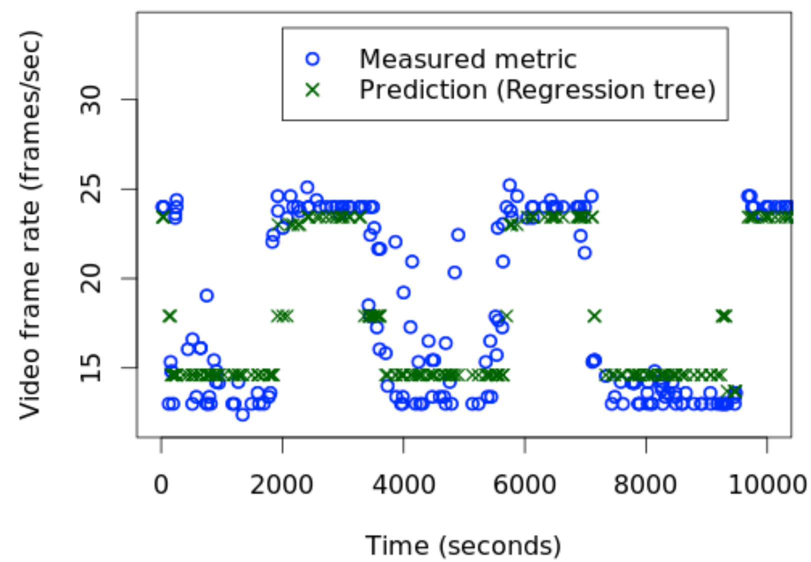


Video frame rate



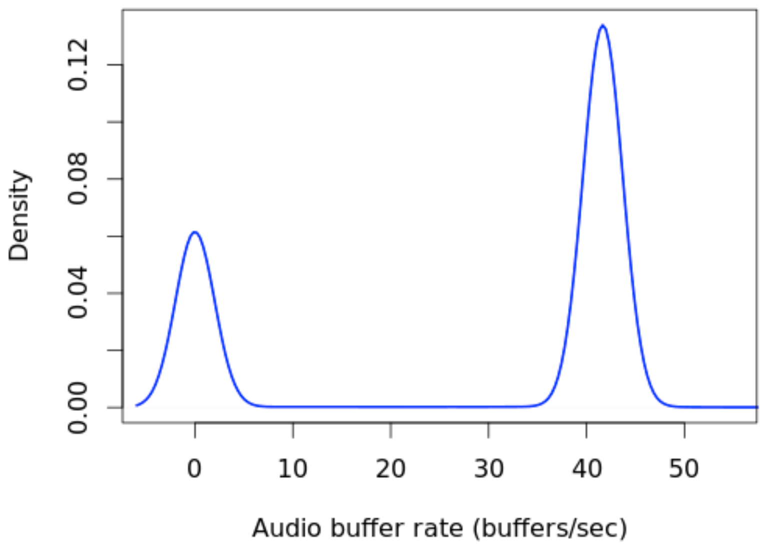


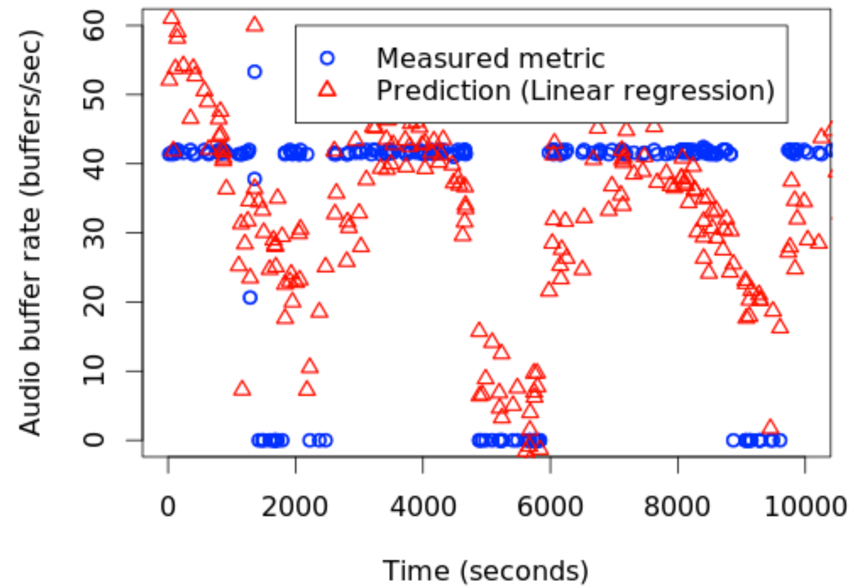
Method	NMAE (%)
Linear regression	12
Regression tree	11



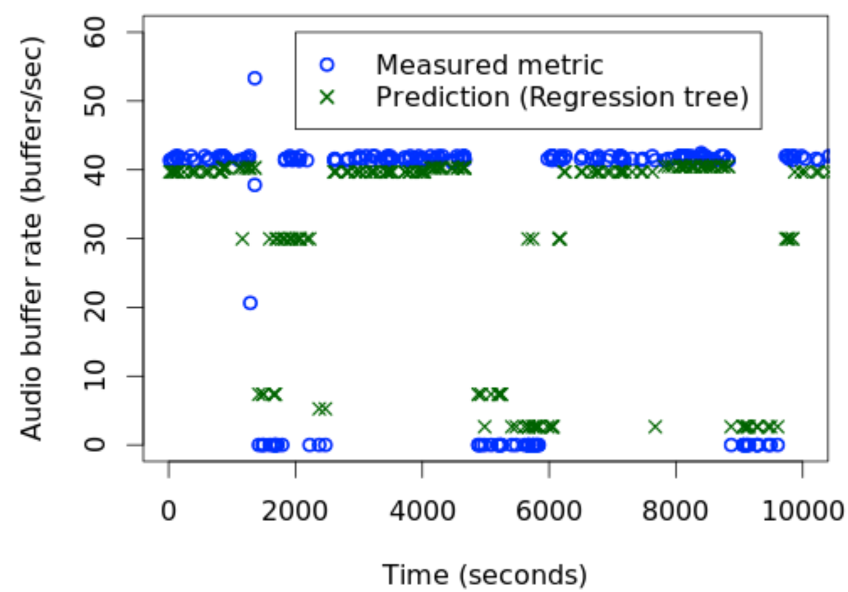
- Y – bimodal distribution
- Both methods provide similar prediction errors

Audio buffer rate



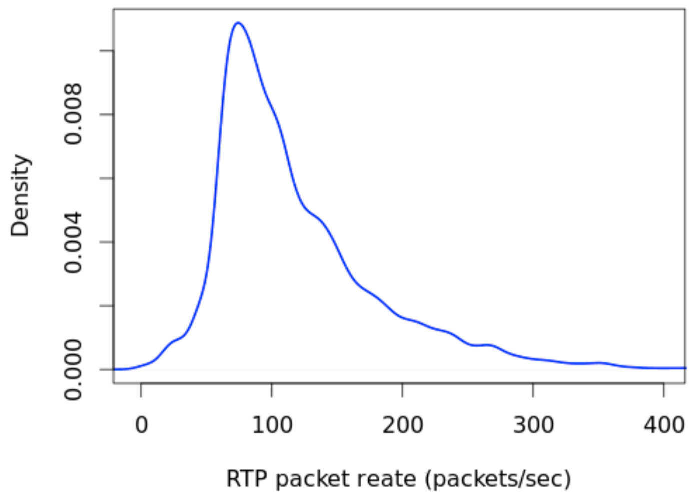


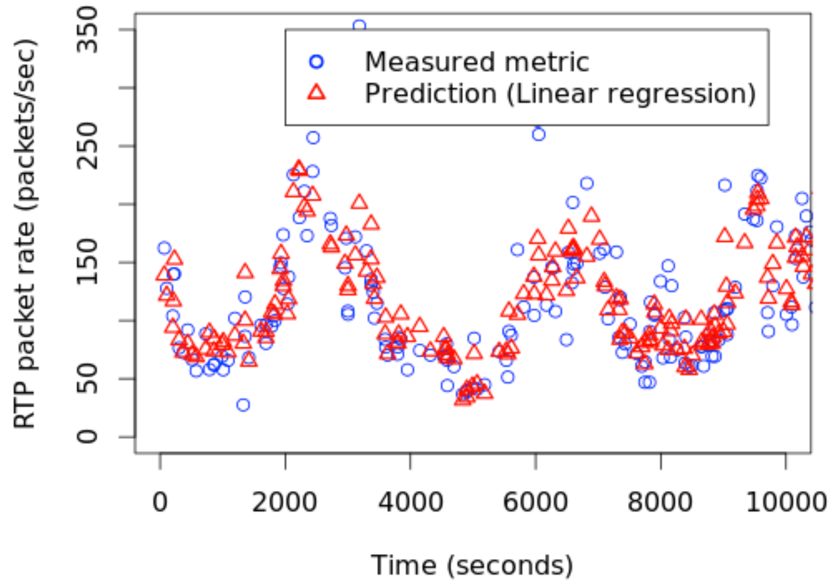
Method	NMAE (%)
Linear regression	41
Regression tree	19



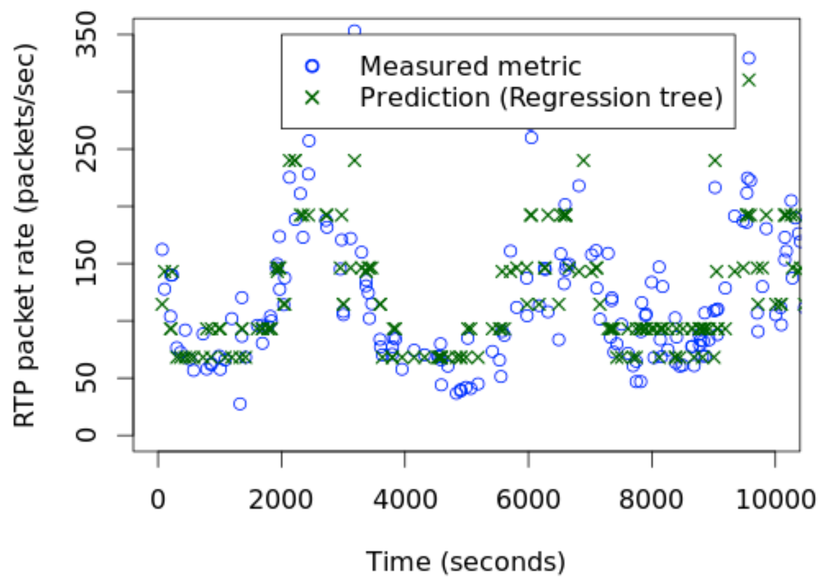
- Y – bimodal distribution
- Regression tree outperforms least-square linear regression

RTP packet rate





Method	NMAE (%)
Linear regression	15
Regression tree	19



- Y – wider spread distribution
- Least-square linear regression outperforms regression tree

Evaluation results – periodic-load trace

Device statistics	Regression method	NMAE (%)		
		Video	Audio	RTP
X_proc	Linear regression	26	59	39
	Lasso regression	23	63	35
	Regression tree	23	61	36
	Random forest	22	60	34
X_sar	Linear regression	12	41	15
	Lasso regression	16	51	17
	Regression tree	11	19	19
	Random forest	**6**	**0.94**	**15**

Lessons learned

- Appropriate models depend on the service metrics
- It is feasible to accurately predict client-side metrics based on low-level device statistics
  - NMAE below 15% across service-level metrics and traces
- Preprocessing of X is critical
  - X_sar outperforms X_proc
  - Significant improvement of prediction accuracy
- There is a trade-off between computational resources vs. prediction accuracy
  - Random forest vs. linear regression

Automatic feature selection

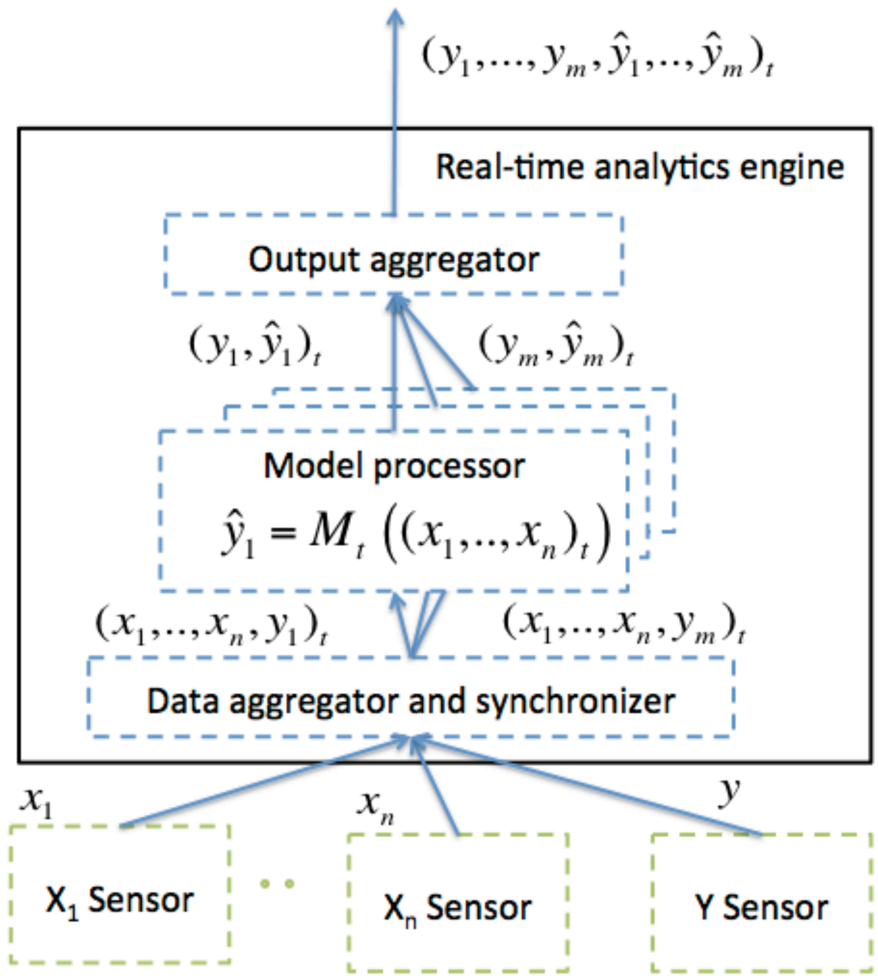
- Exhaustive search is infeasible
  - Requires $O(2^p)$ training executions ($p = \sim5000$)
- Forward stepwise feature selection
  - Heuristic method $O(p^2)$ training executions
  - Start with an empty set
  - Incrementally grows the feature set with one feature at a time that minimizes a specific evaluation metric
    - Selected metric is the cross-validation error

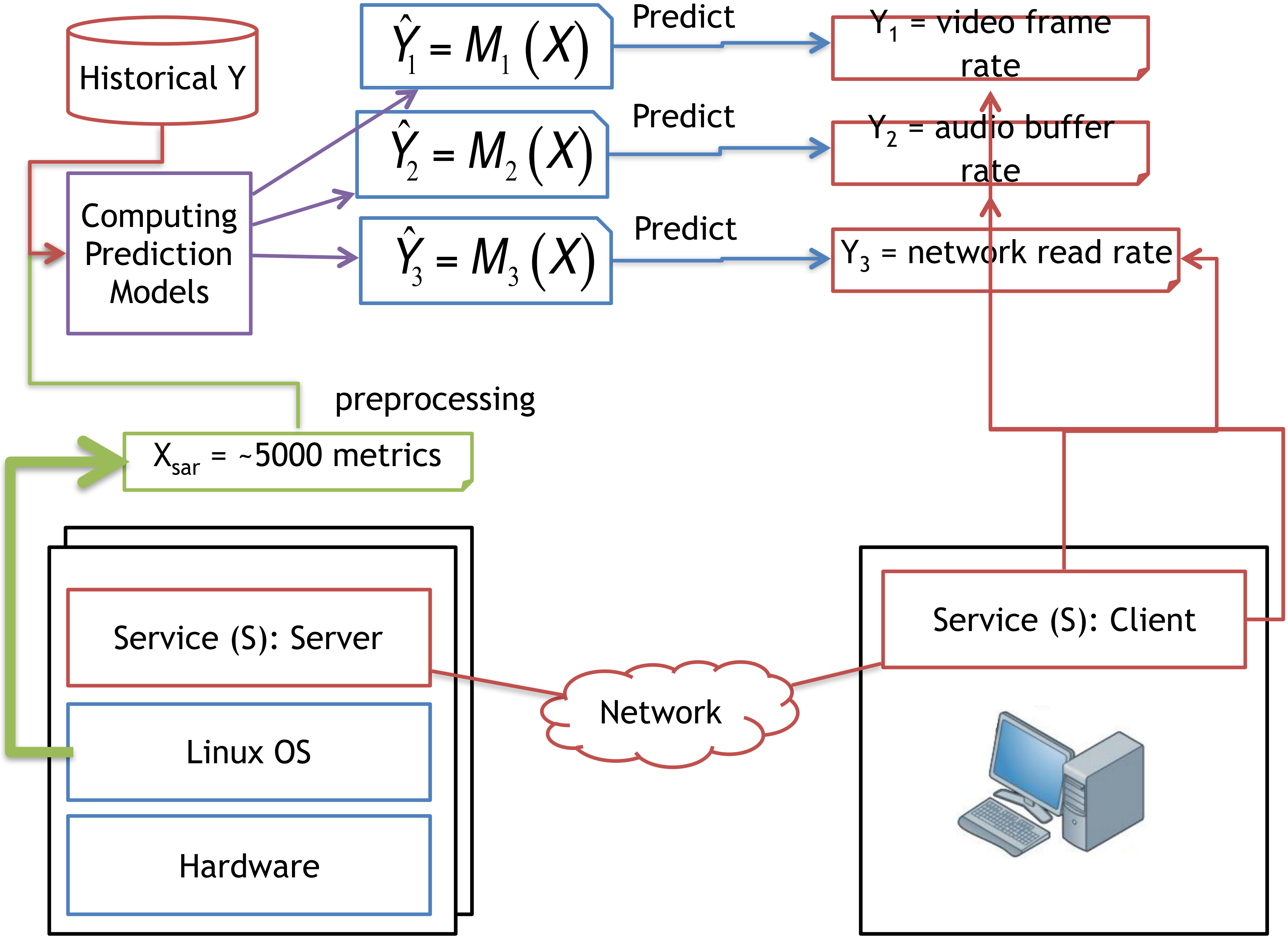
Random forest on different feature sets

Trace	Feature set	Video		Audio	
		NMAE (%)	Training (secs)	NMAE(%)	Training (secs)
Periodic-load	Full	12	> 50000	32	> 70000
	Automatically optimized feature set	**6**	862	**22**	1600
Flash-load	Full	8	> 55000	21	> 75000
	Automatically optimized feature set	**4**	778	**15**	1750

* Learning with the optimized feature set significantly improve prediction accuracy and reduce training time

Real-time Analytics Engine

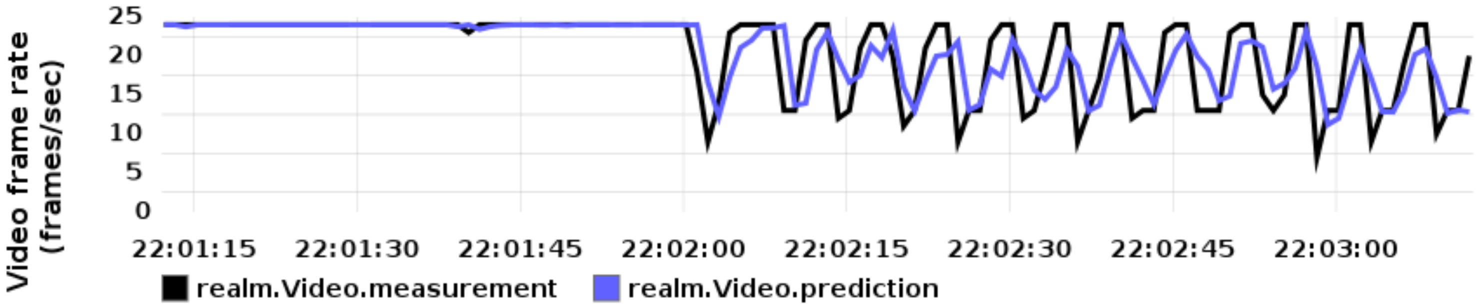




Historical Y
Computing Prediction Models
$\hat{Y}_1 = M_1(X)$
$\hat{Y}_2 = M_2(X)$
$\hat{Y}_3 = M_3(X)$
Predict
Predict
Predict
Y_1 = video frame rate
Y_2 = audio buffer rate
Y_3 = network read rate
preprocessing
X_{sar} = ~5000 metrics
Service (S): Server
Linux OS
Hardware
Network
Service (S): Client

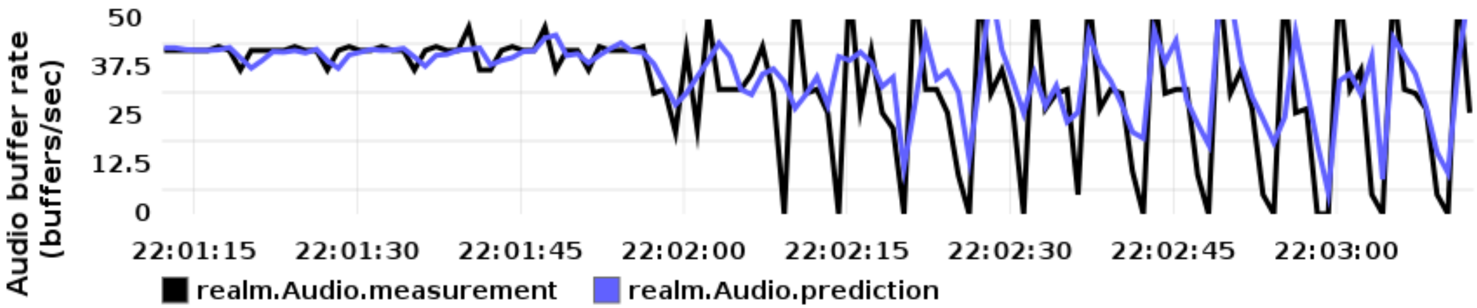
Recorded Demo

Real-time Predictions of Service Metrics from Device Statistics



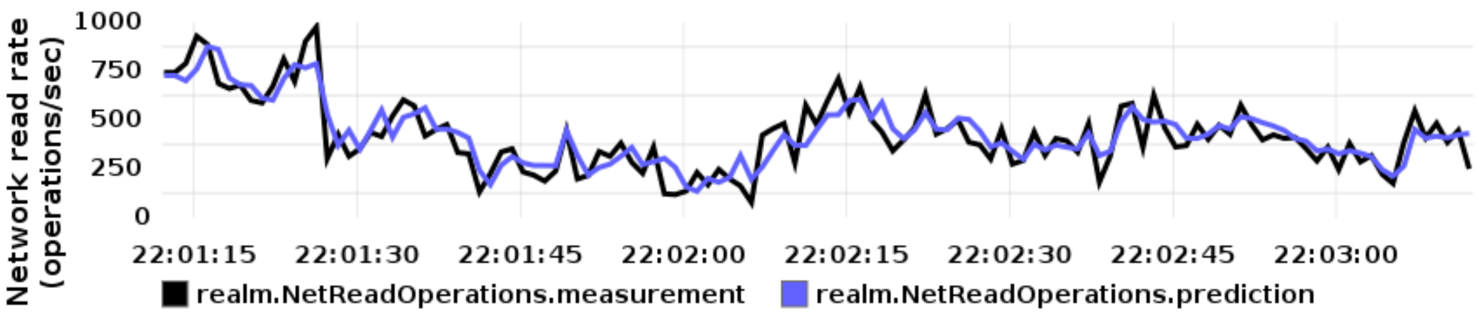
Normalized Mean Absolute Error (last 5 minutes)

4.43 %



Normalized Mean Absolute Error (last 5 minutes)

13.73 %



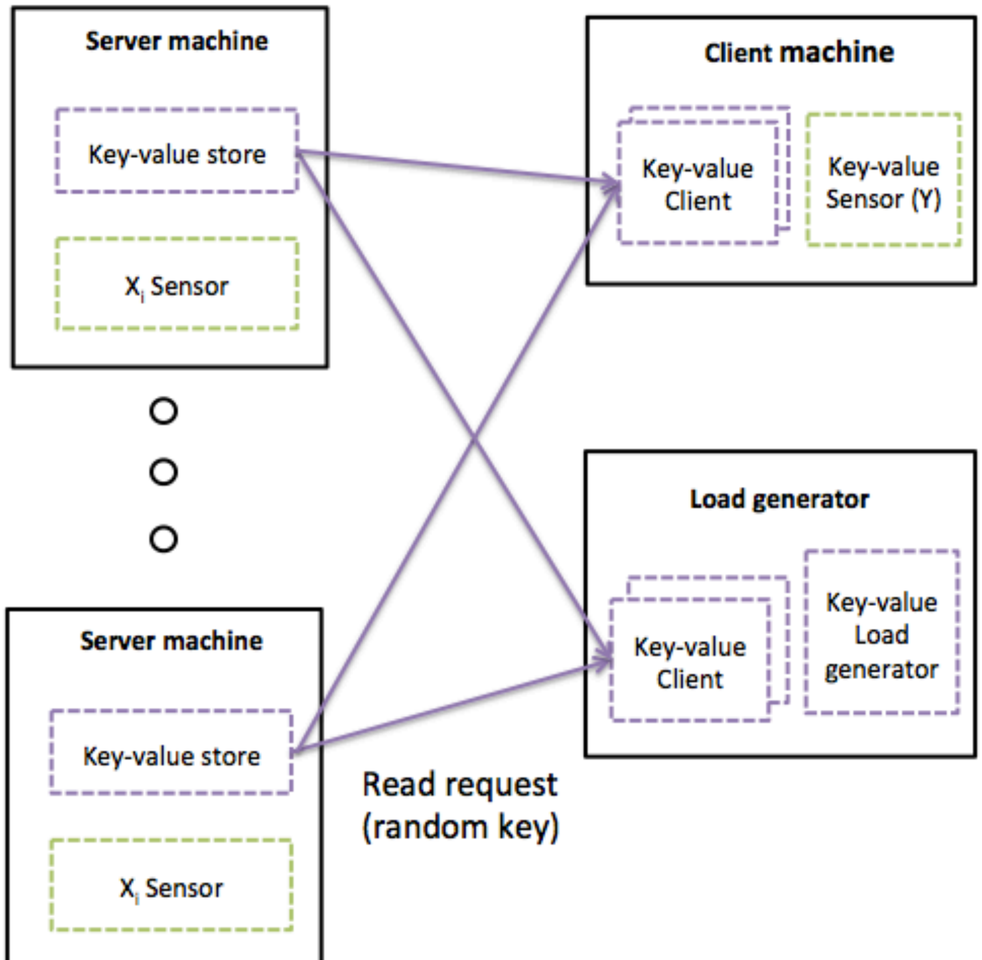
Normalized Mean Absolute Error (last 5 minutes)

18.77 %

Recap

- Design and develop experimental testbeds
- Run experiments with various load patterns
- Collected and published traces
- Machine learning model assessment
  - Batch learning on traces
  - Online learning on traces
  - Real-time learning on live statistics
- Design and develop real-time analytics engine
- Implement a prototype for real-time learning on live statistics

Key-value store (Voldemort)



Questions?