Smart Home Network Management with Dynamic Traffic Distribution

Chenguang Zhu

Xiang Ren

Tianran Xu

Motivation

Motivation – Per Application QoS

- In small home / office networks,
  applications compete for limited bandwidth
- high bandwidth consumption applications can be disruptive
  - Eg. bitTorrent

- To ensure fairness,
  different application flows should be given different priorities
  - Eg. High priority for important Skype meeting
  - Eg. Low priority for bitTorrent download

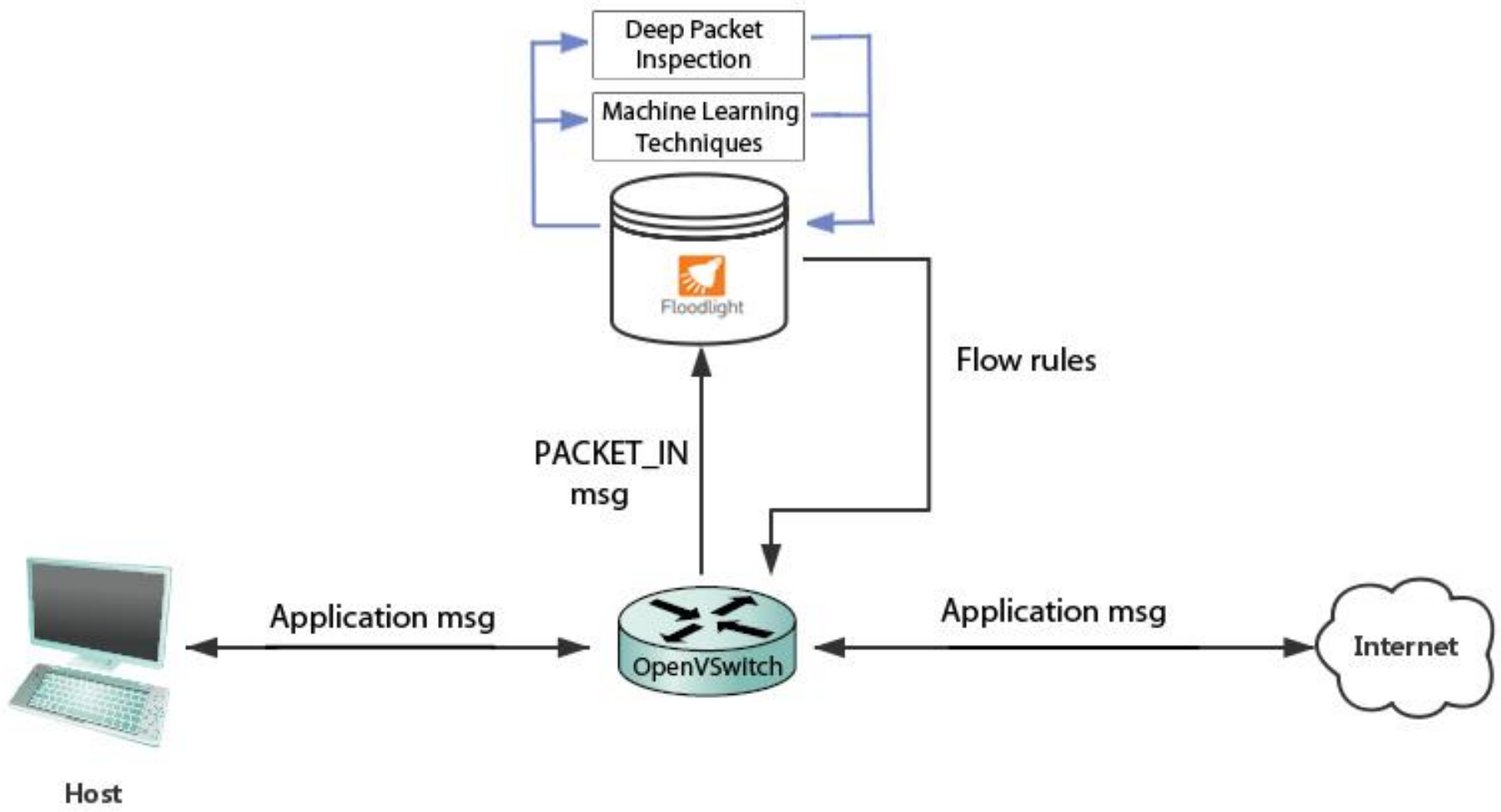
- Need traffic adjustment based on flow types

Motivation – Per Application QoS

- Flow identification is difficult in traditional networks
- SDN allows novel flow identification techniques
  - Deep packet inspection
  - Machine learning based techniques
- Use flow rules to easily adjust traffic

System Design

Design – System Overview



Flow Identification – Commonly Used Techniques

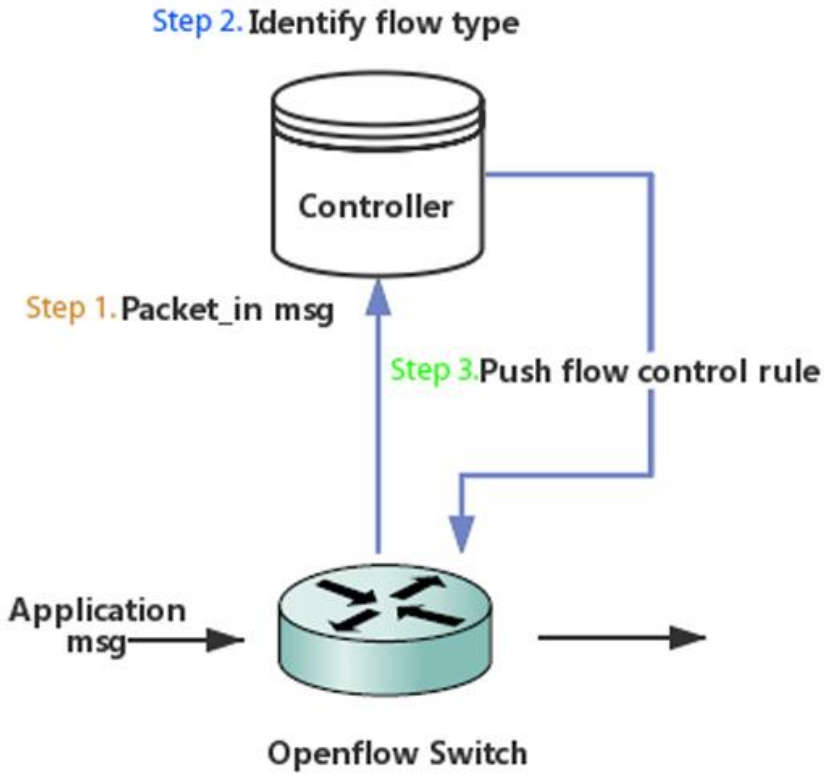
- Shallow packet inspection
  - Inspect packet header, eg. port-number, protocol
  - Low accuracy, application circumvention
- Deep packet inspection
  - Inspect data part of a packet, high accuracy
  - Sometimes maintain a big database of packet features
  - Frequently update rules for new applications

Flow Identification – Machine Learning

- Machine learning based-techniques <<< We focus on this one
  - Novel techniques
  - Cross-disciplinary
  - Interesting experiments
    - eg. Clustering vs classification algorithms

Design - Traffic Adjustment

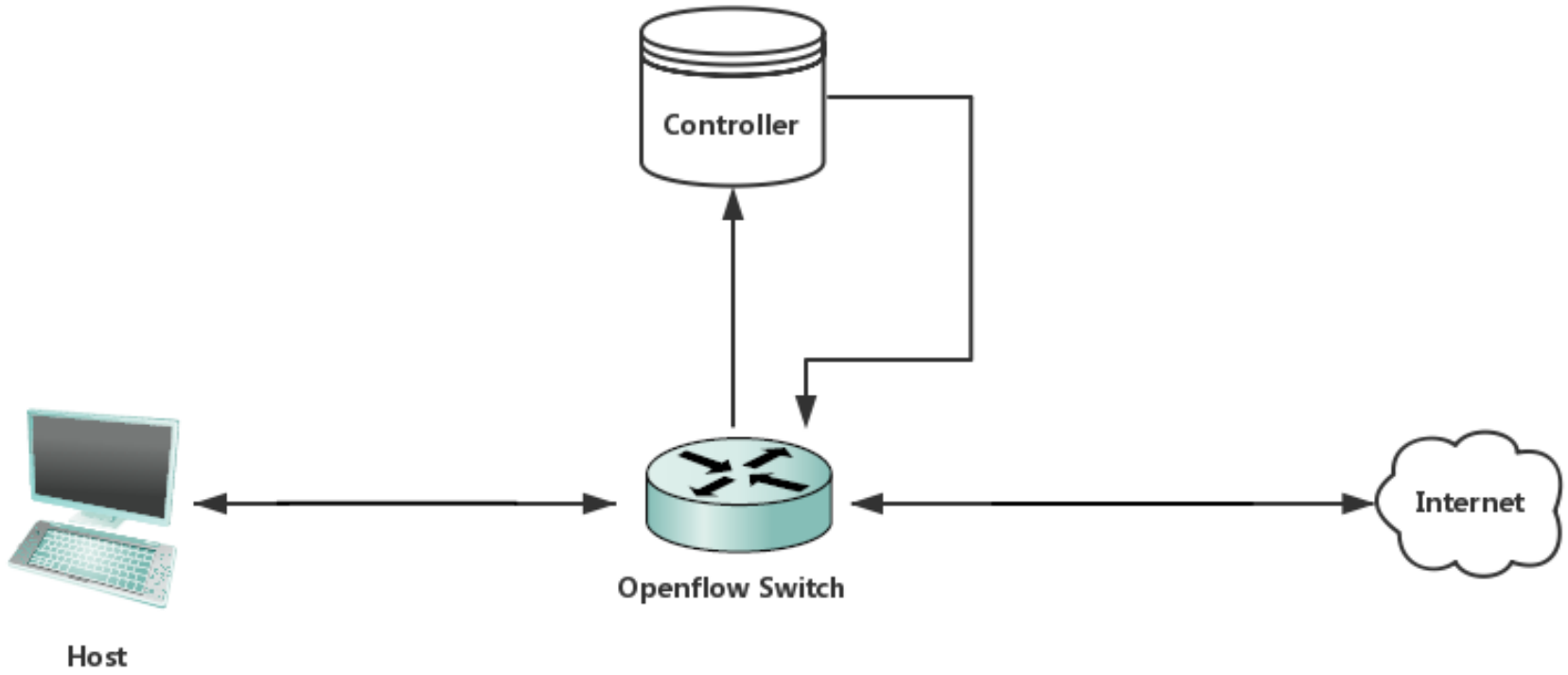
- Assign different priority based on flow type



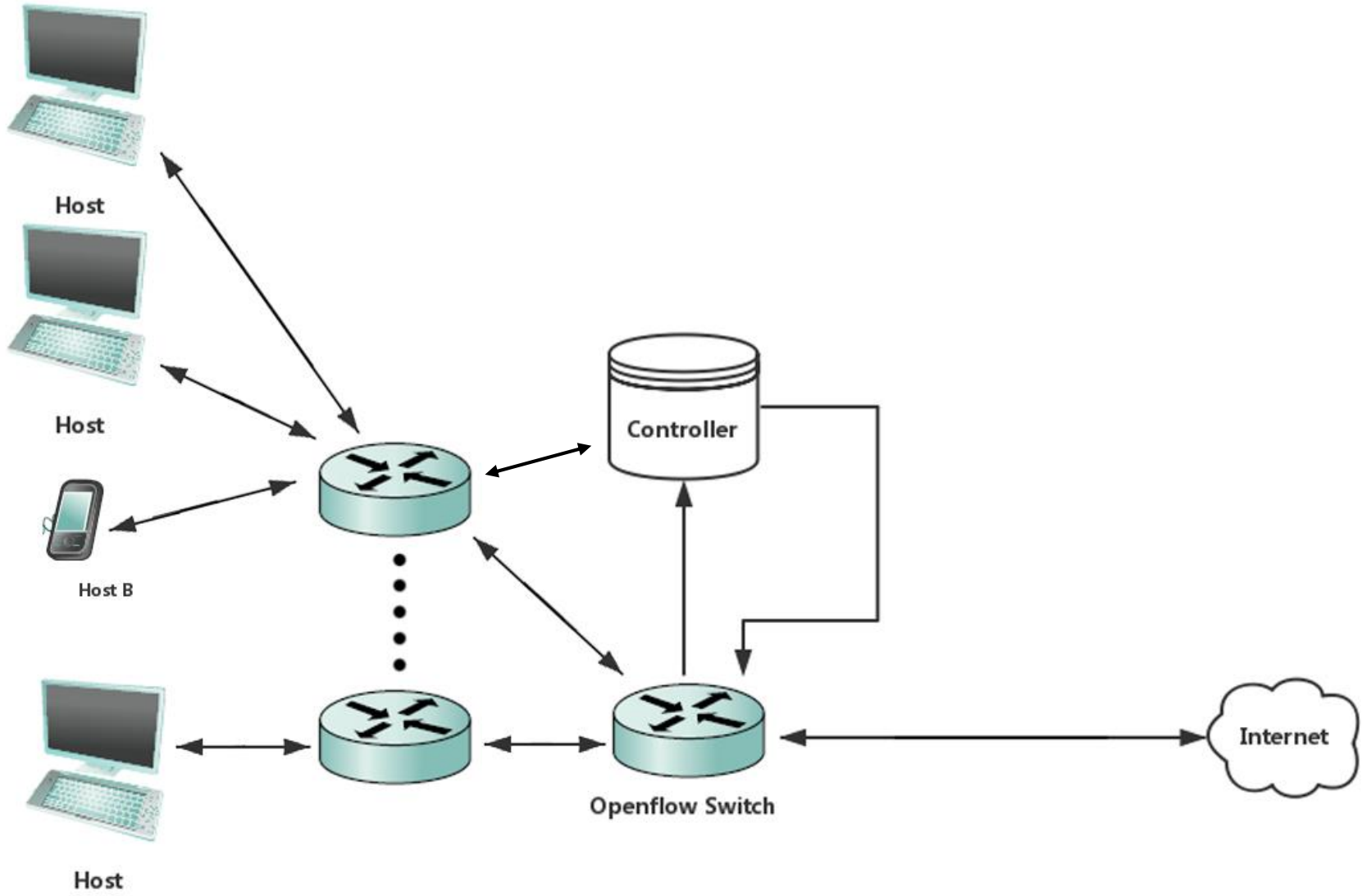
Implementation

Floodlight + Mininet + OpenVSwitch

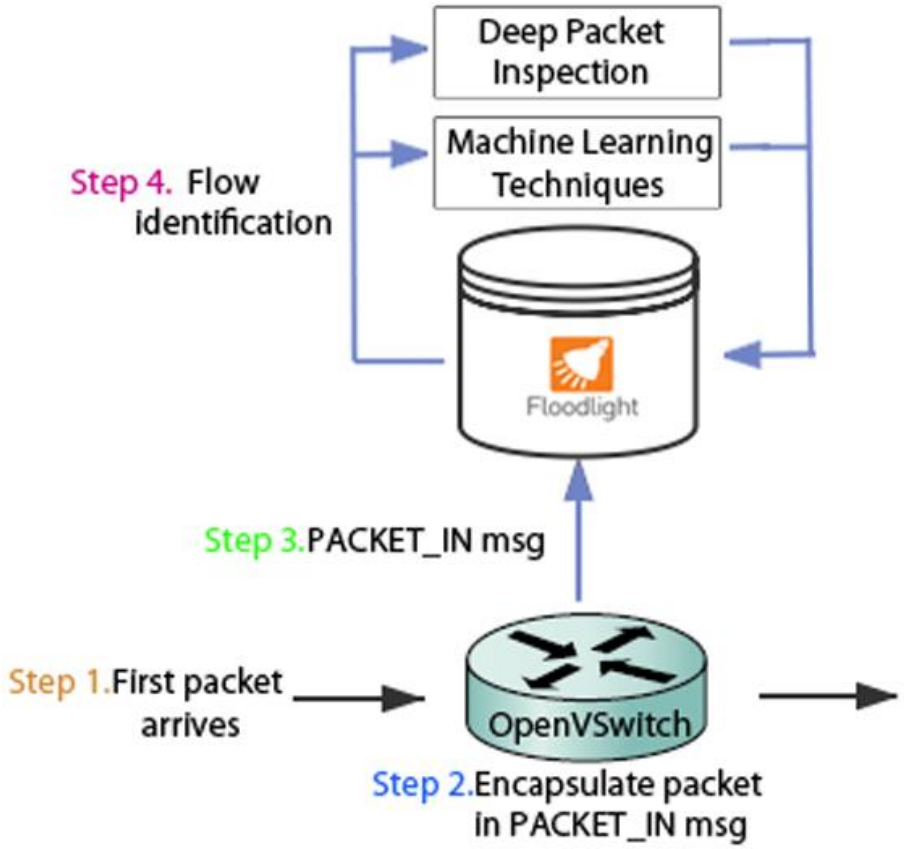
Implementation –Simple Test Topology



Implementation –Realistic Topology



Implementation – Packet Arrival and Identification



Implementation – Deep Packet Inspection

- Inspects data part of a packet
- Use simple rules to identify packet type

Protocol	Data part features
HTTP	contains ‘GET’ ‘DELETE’ ‘POST’ ‘PUT’ …
SSH	start with ‘SSH-’
OpenVPN	first two bytes stores packet length – 2
…	…

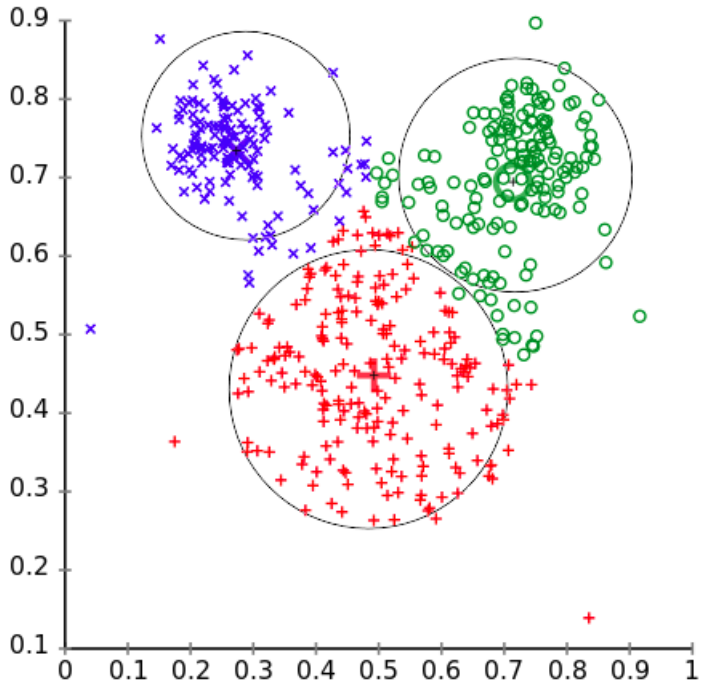
Implementation – Machine Learning Techniques

- Clustering vs Classification
- Clustering:
  - Use K-Means algorithm
- Classification:
  - Use SVM algorithm

Clustering – K-Means

- groups data points into k clusters,

  each point belongs to the cluster with the nearest mean

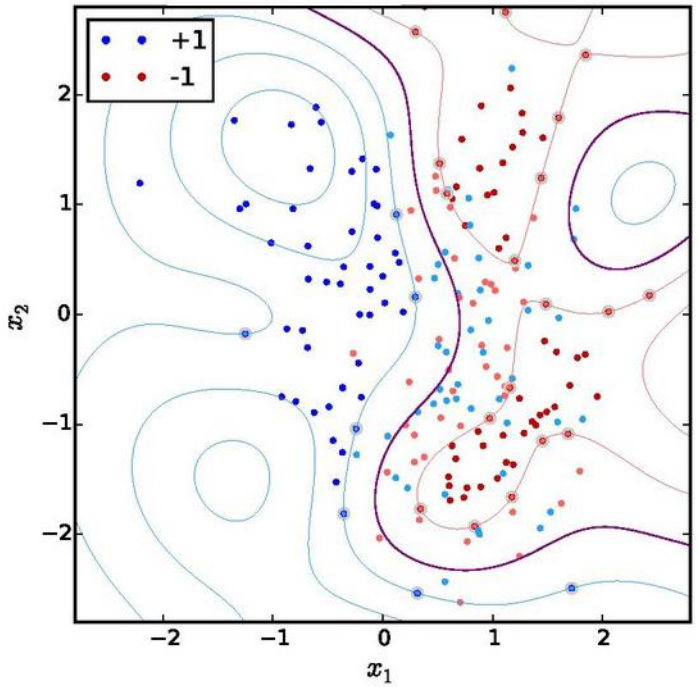


- Source: https://en.wikipedia.org/wiki/K-means_clustering

Classification - SVM

- assigns data points into categories,

  based on data vectors nearest to the category boundaries



- Source: https://en.wikipedia.org/wiki/Support_vector_machine

Dataset Selection

- Publically available research traces
  - eg. waikato traces (http://wand.net.nz/wits/catalogue.php)
  - Pros: representative traffic workloads
  - Cons: too complex, hard to label packet type
- Self collected traces
  - Self generated packets, captured on WireShark
  - Easy to label

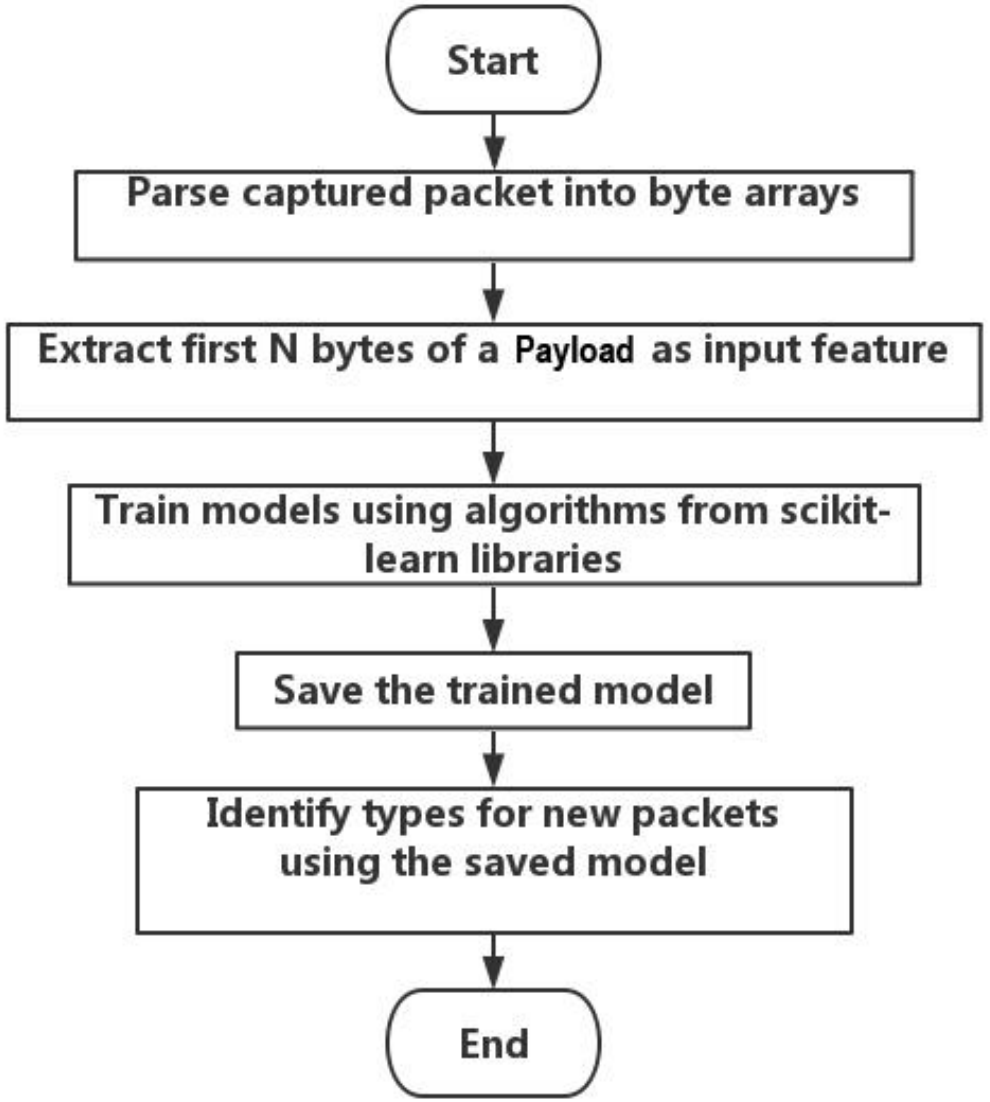
Feature

- Commonly used features from research literature

Features
Total number of packets per flow
Flow duration
Packet lengths statistic (min, max, mean, std dev.) per flow
Payload lengths
Payload content (We use first N number of bytes of payload as feature)
…

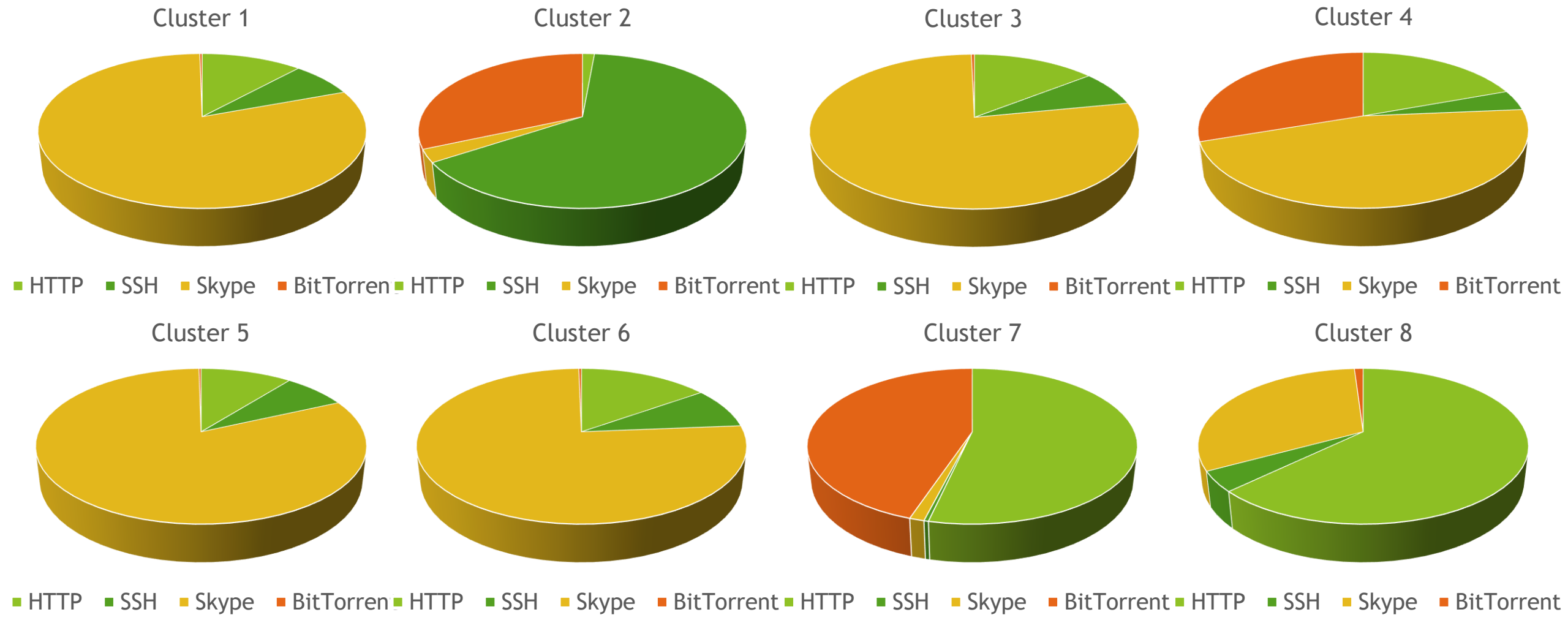
- Source: T. Nguyen and G. Armitage. “A Survey of Techniques for Internet Traffic Classification using Machine Learning” IEEE Communications Surveys and Tutorials 01/2008; 10:56-76.

Machine Learning Based Identification



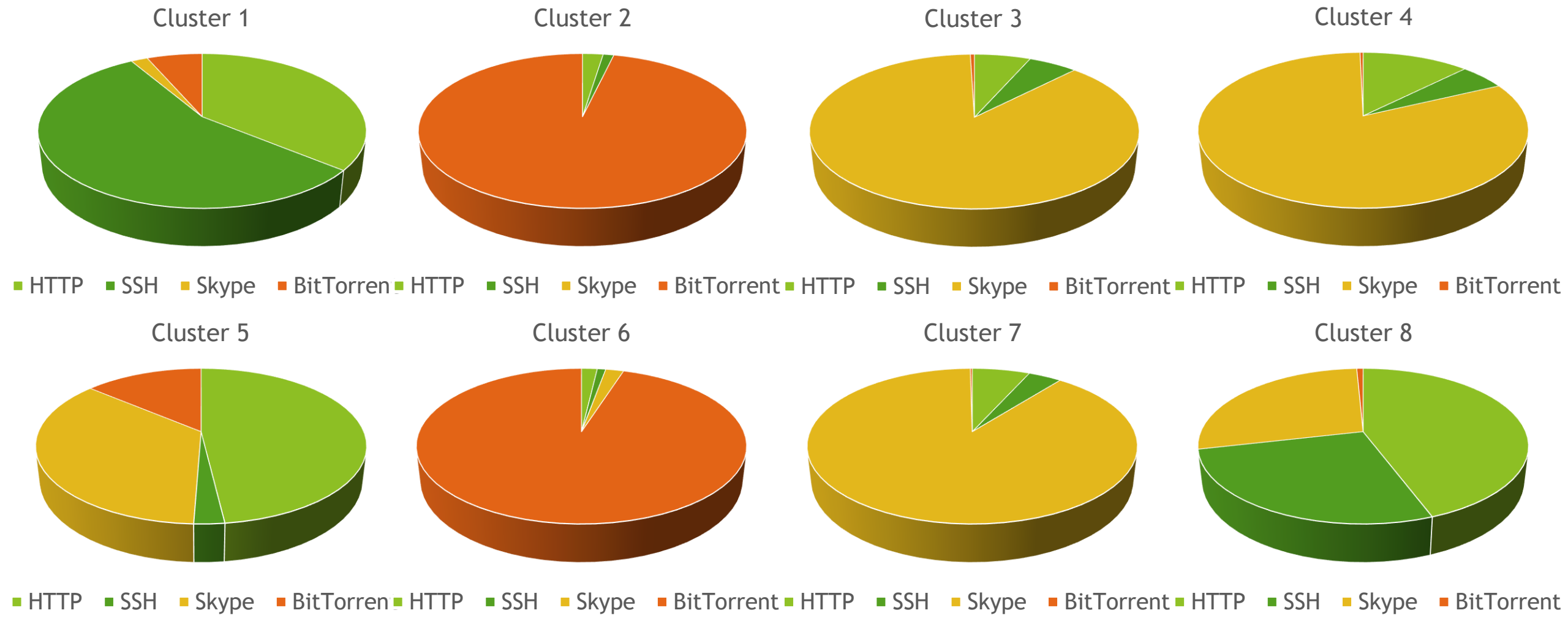
Performance of Identification – K-Means

K-means 2 bytes



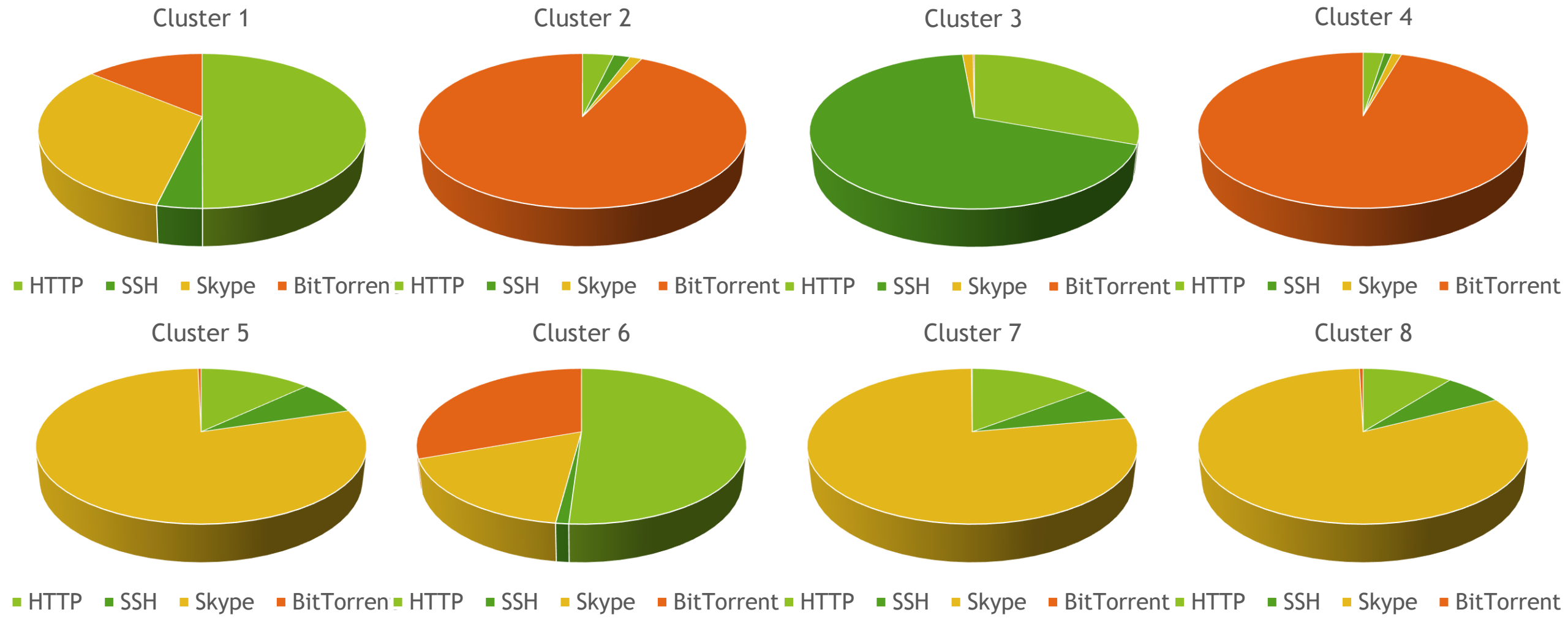
Performance of Identification – K-Means

K-means 3 bytes



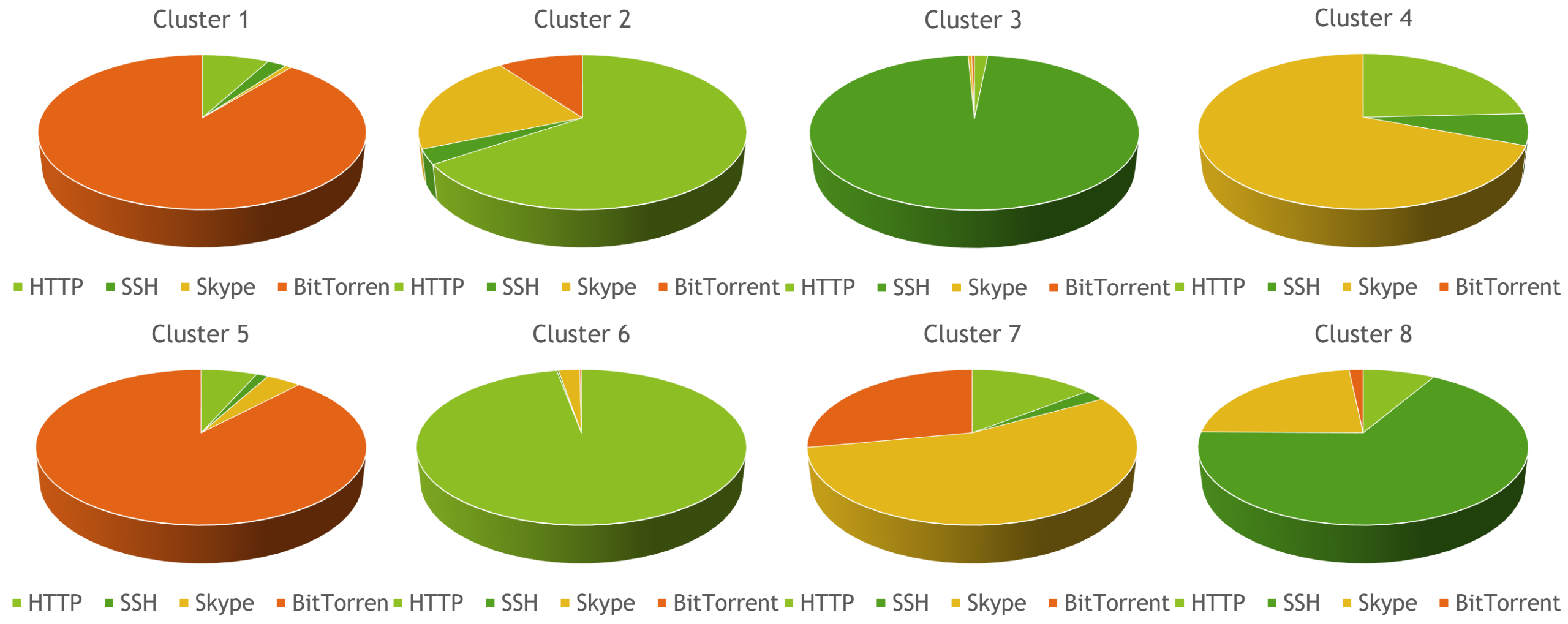
Performance of Identification – K-Means

K-means 4 bytes



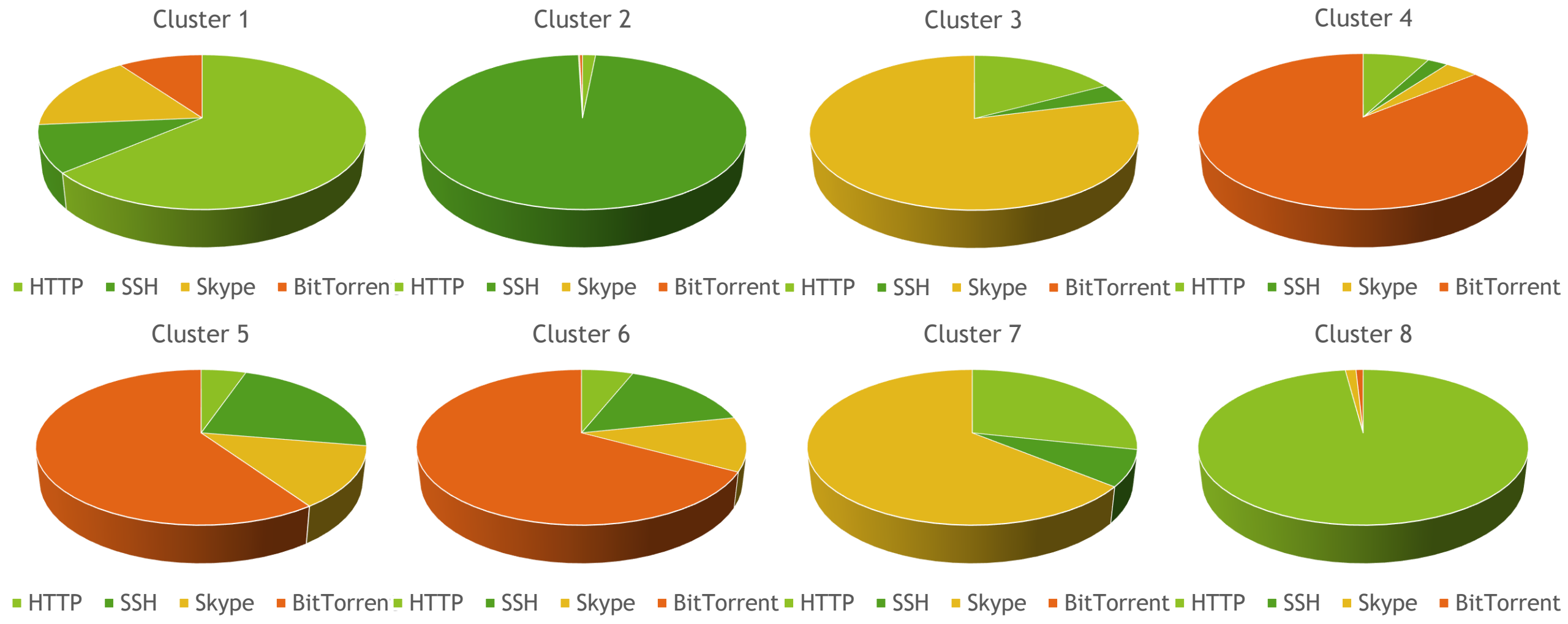
Performance of Identification – K-Means

K-means 8 bytes

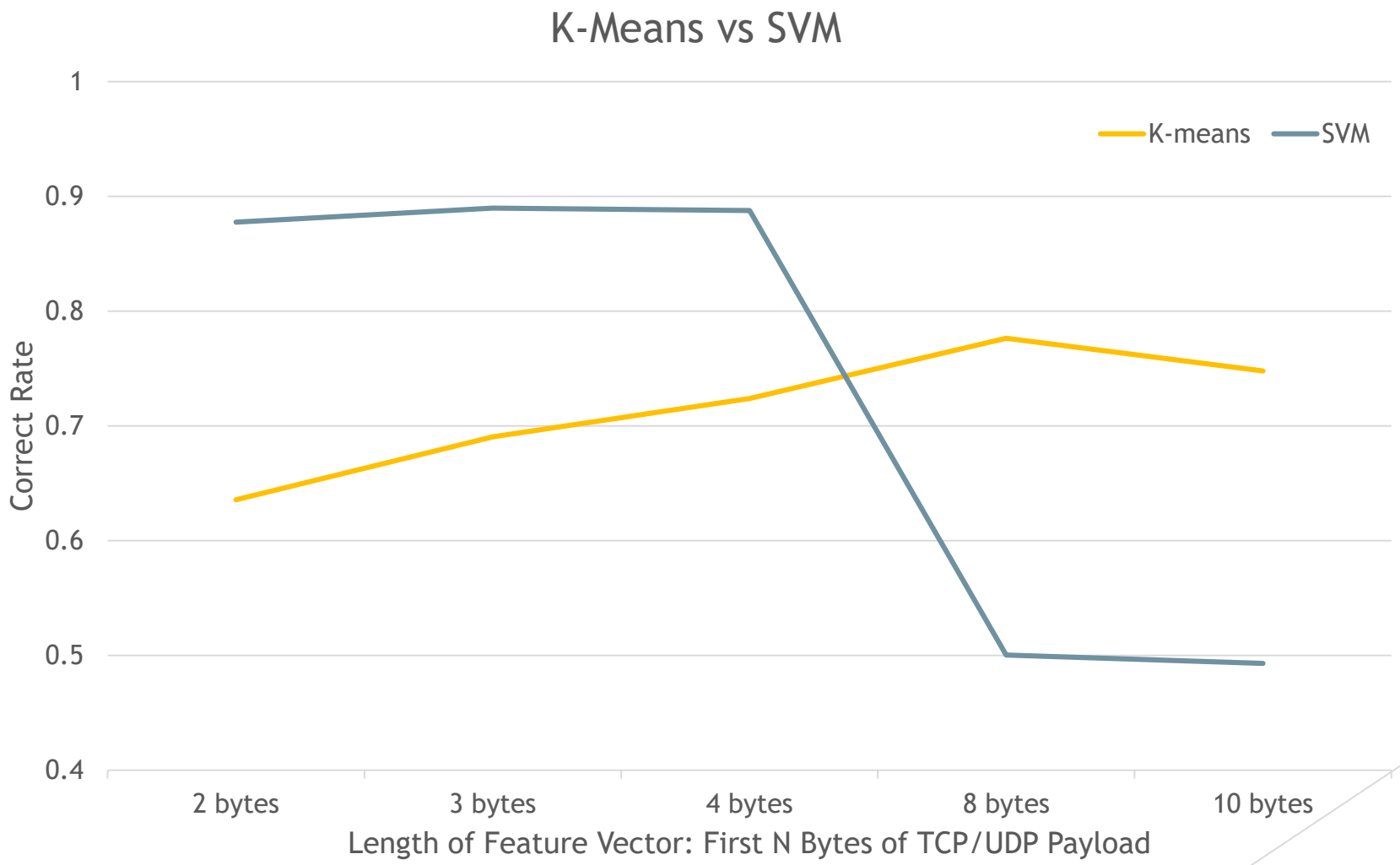


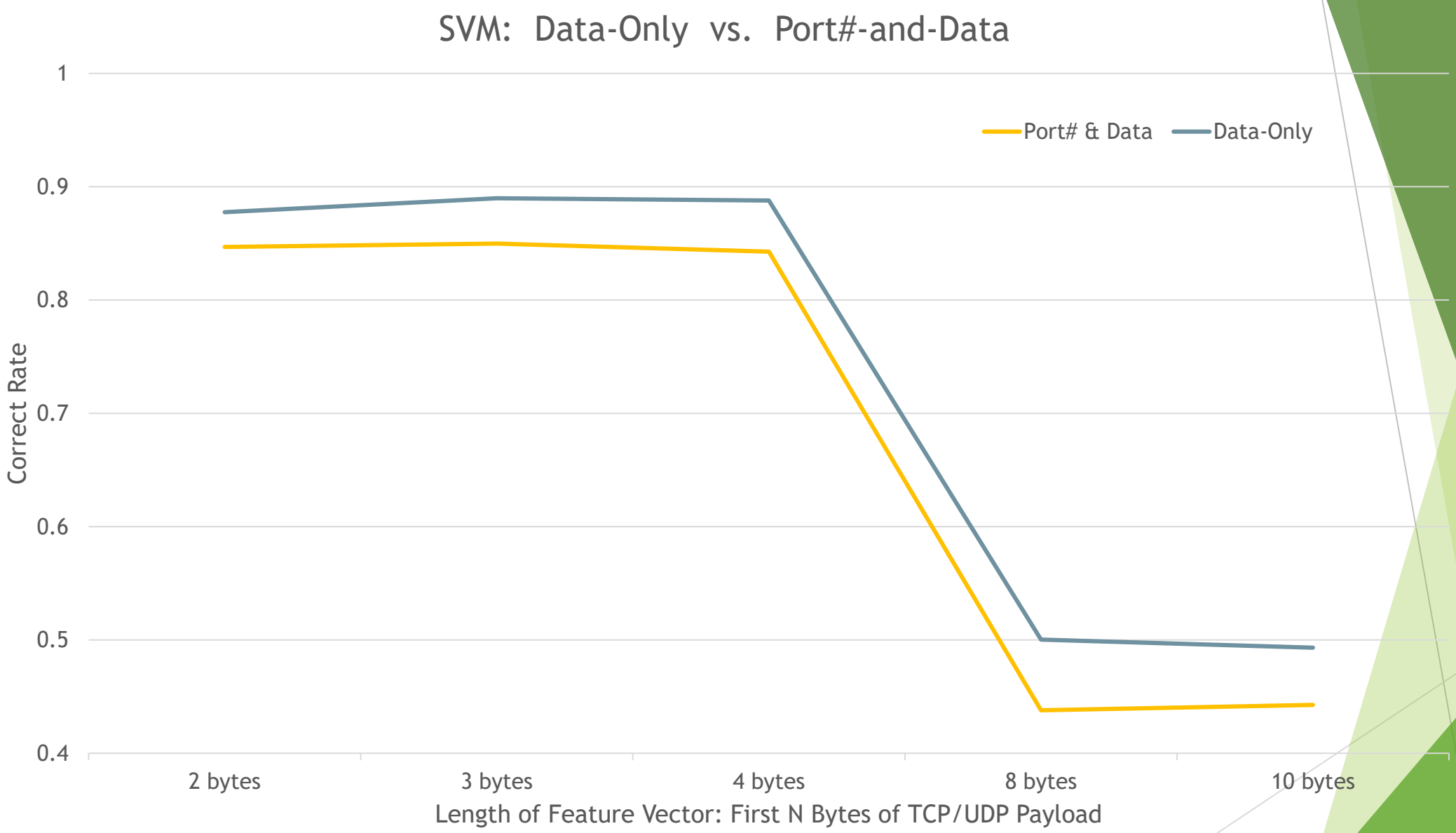
Performance of Identification – K-Means

K-means 10 bytes

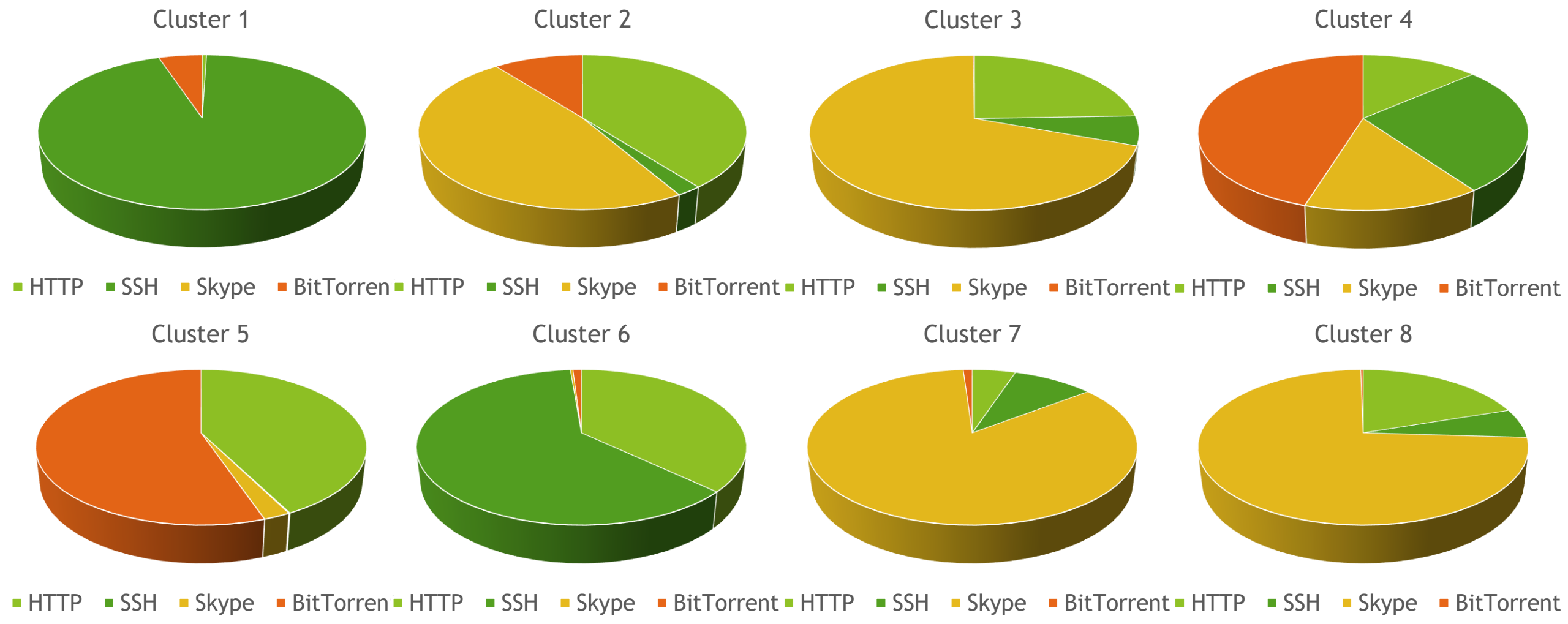


Performance of Identification – Varying Feature Length

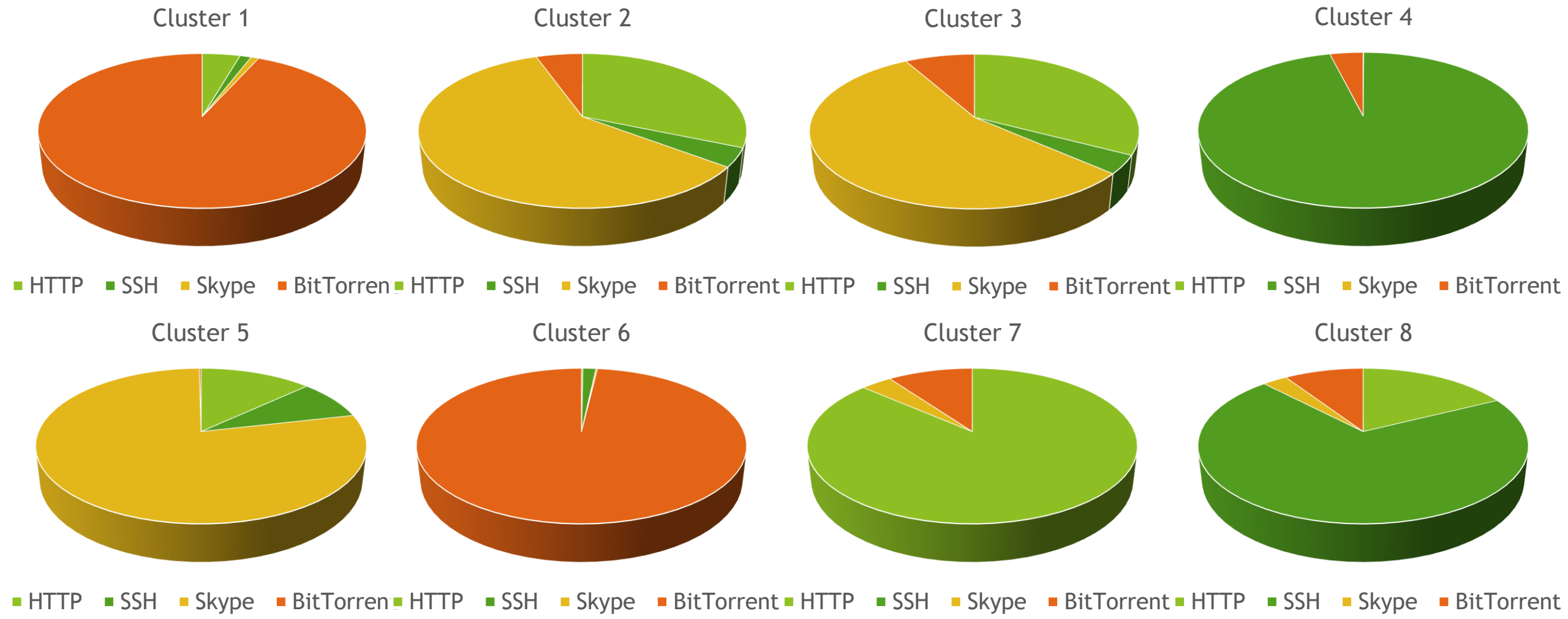



SVM: Data-Only vs. Port#-and-Data
1
0.9
0.8
0.7
0.6
0.5
0.4
Correct Rate
Port# & Data
Data-Only
2 bytes
3 bytes
4 bytes
8 bytes
10 bytes
Length of Feature Vector: First N Bytes of TCP/UDP Payload

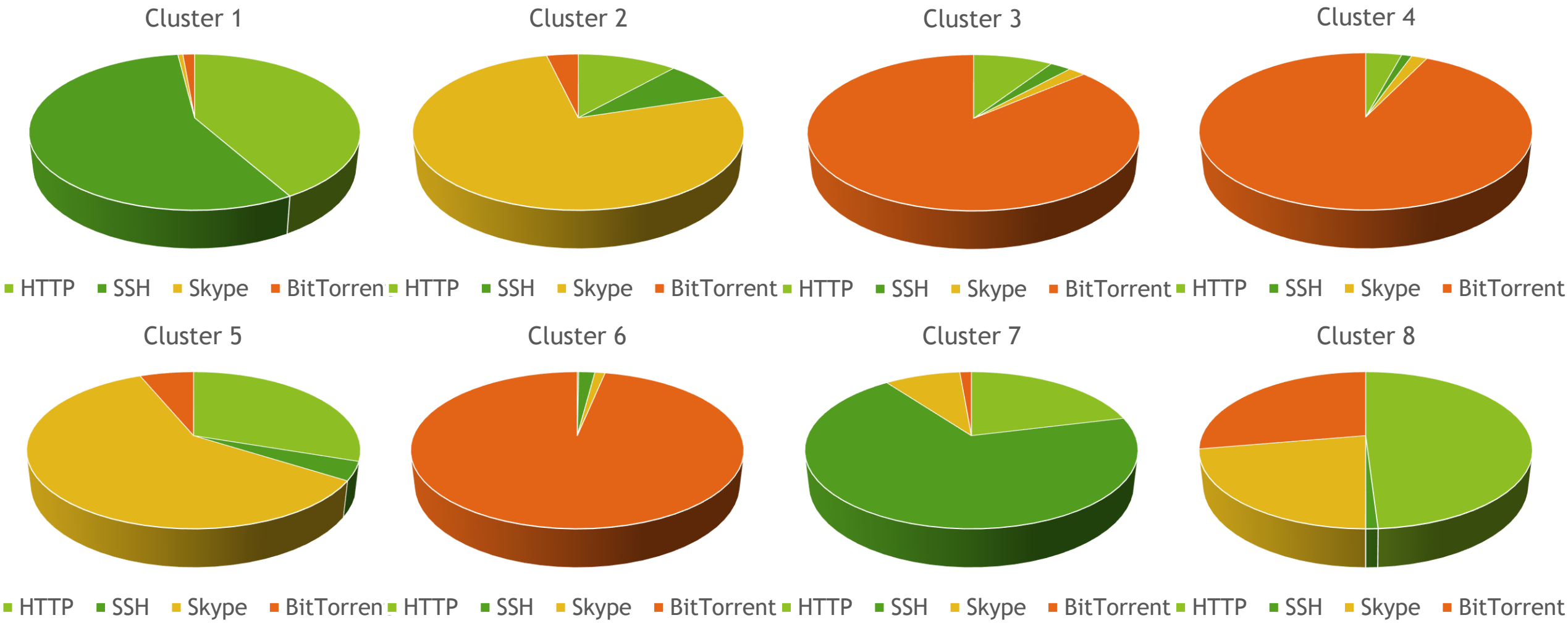
K-means port# + 2 bytes data



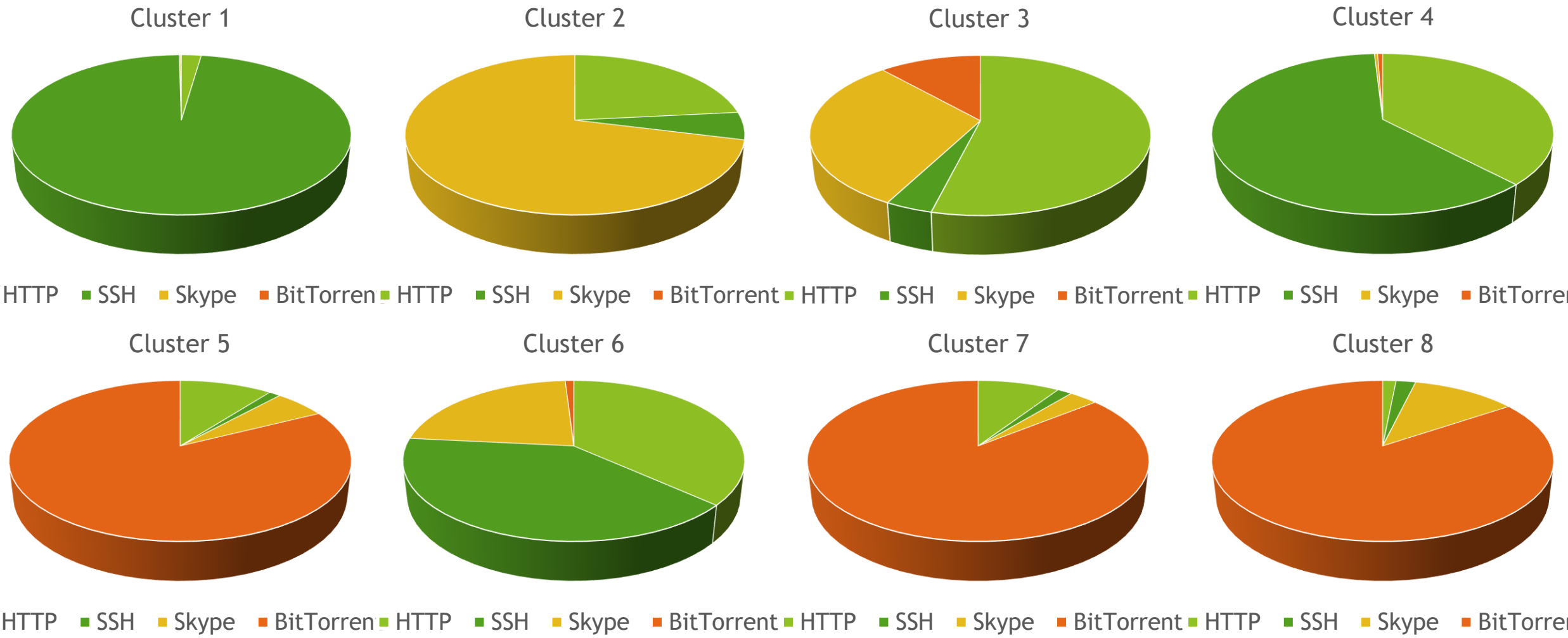
K-means port# + 3 bytes data



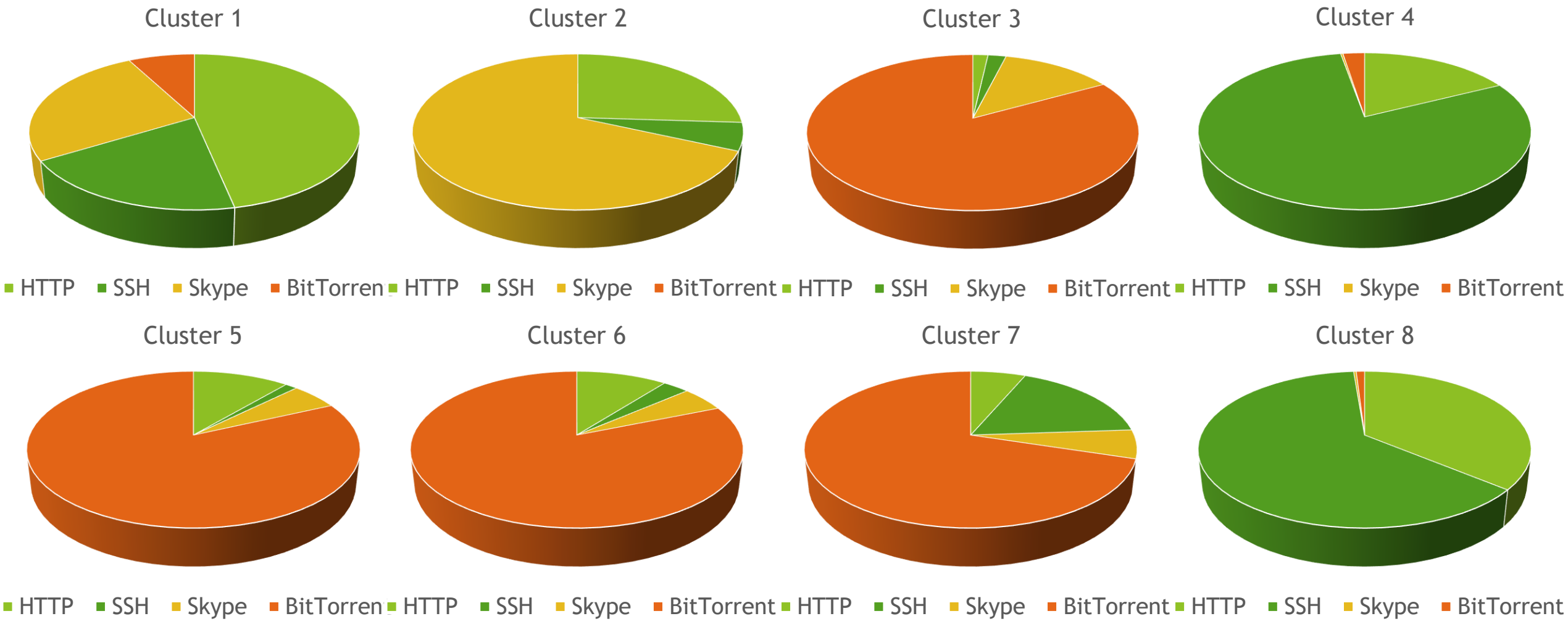
K-means port# + 4 bytes data



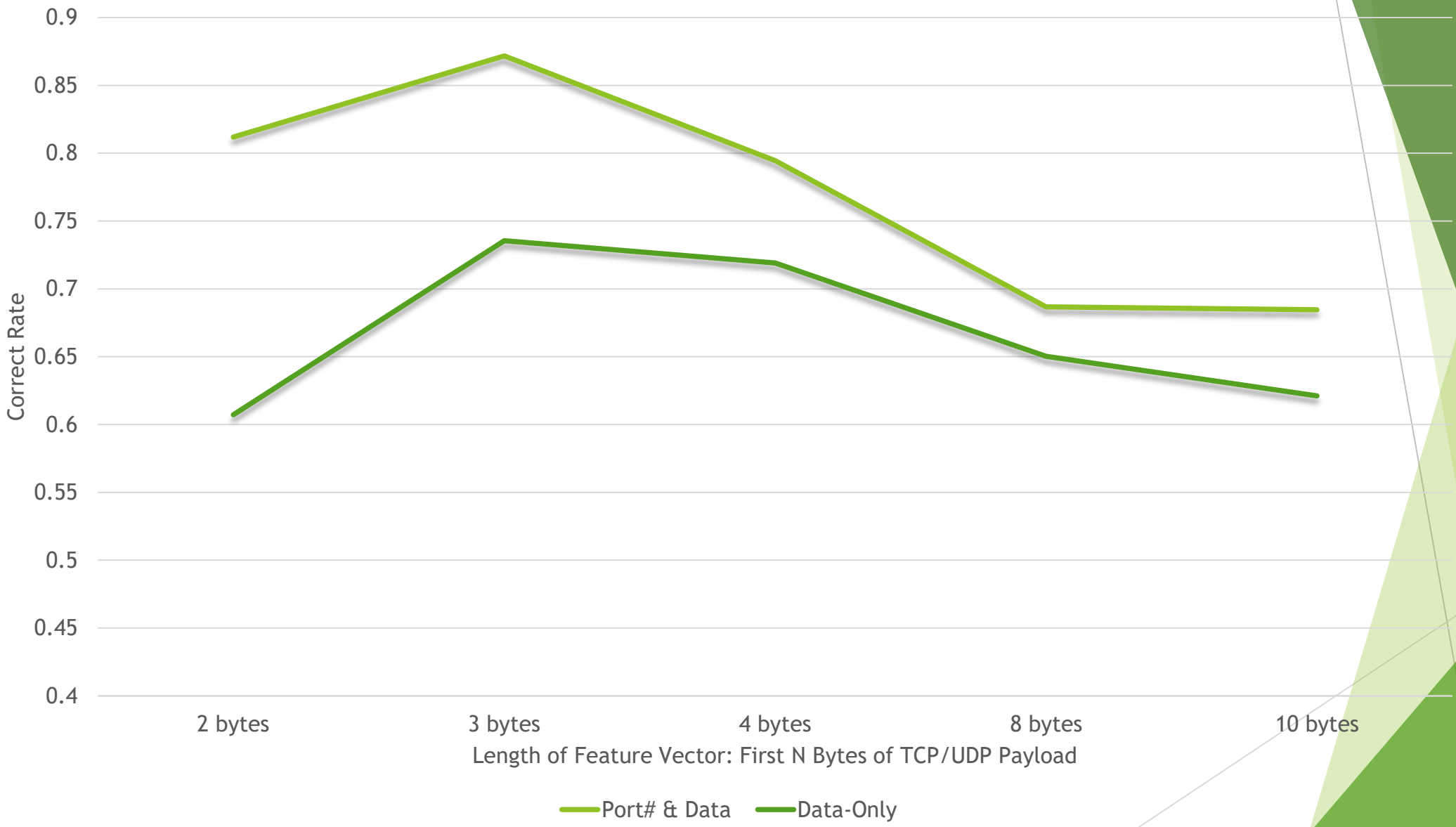
K-means port# + 8 bytes data

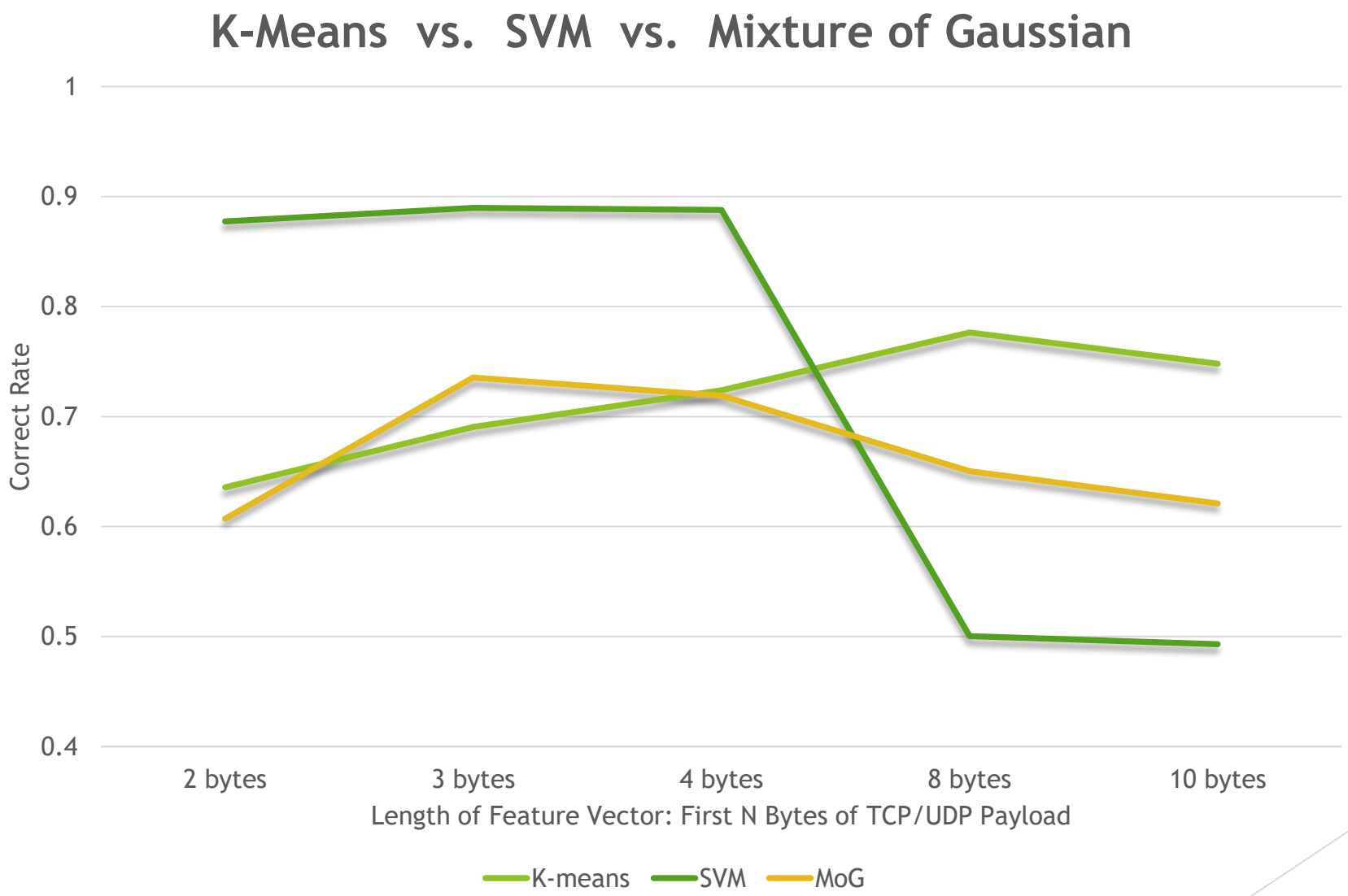


K-means port# + 10 bytes data



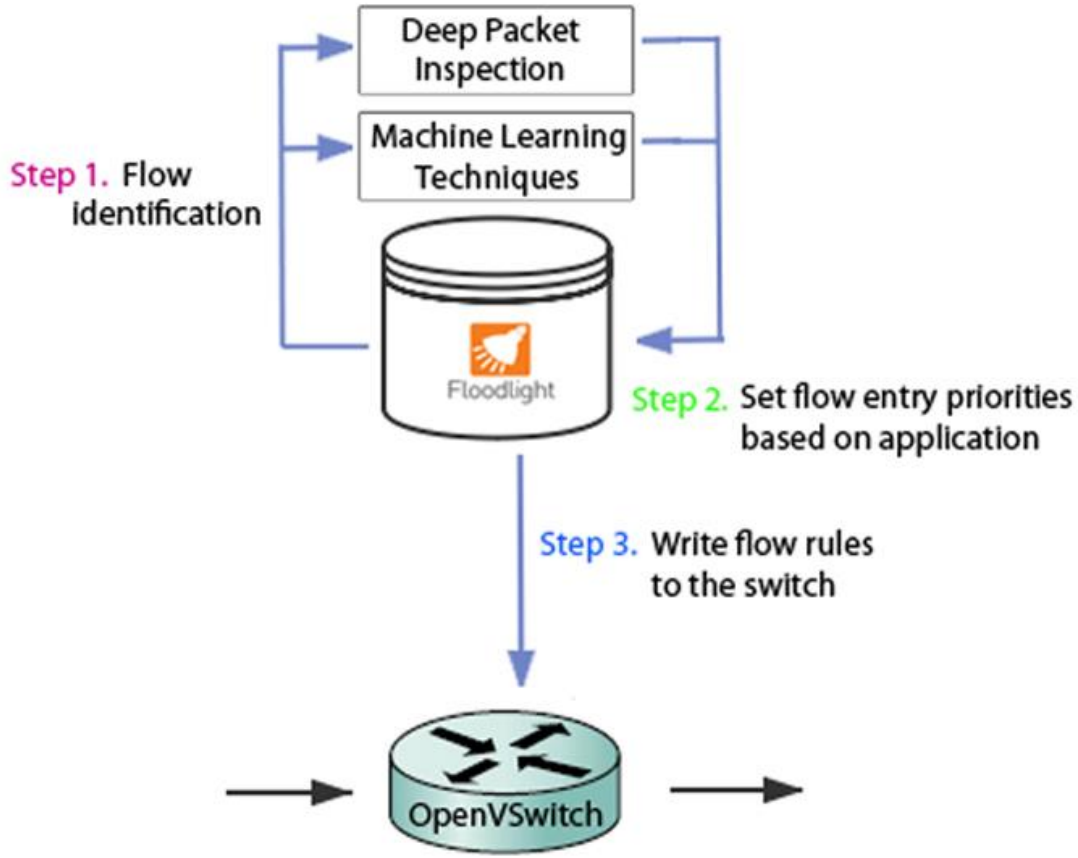
Mixture of Gaussian: Data-Only vs. Port#-and-Data




K-Means vs. SVM vs. Mixture of Gaussian
1
0.9
0.8
0.7
0.6
0.5
0.4
Correct Rate
2 bytes
3 bytes
4 bytes
8 bytes
10 bytes
Length of Feature Vector: First N Bytes of TCP/UDP Payload
K-means
SVM
MoG

Performance of Identification – Varying Sample Size

Implementation – Traffic Adjustment



- Next step, direct flows through paths with different bandwidth for QoS

Implementation – Flow Rules

Cookie	Table	Priority	Match	Apply Actions	Write Actions	Clear Actions	Goto Group	Goto Meter	Write Metadata	Experimenter	Packets	Bytes	Age (s)	Timeout (s)
0	0x0	10	eth_type=0x0x800 ip_proto=0x6 tcp_src=44971 tcp_dst=80 ipv4_src=192.168.3.2 ipv4_dst=74.125.226.81	actions:output=2	n/a	n/a	n/a	n/a	n/a	n/a	5	382	68	600
0	0x0	100	eth_type=0x0x800 ip_proto=0x11 udp_src=19233 ipv4_src=192.168.3.2 ipv4_dst=74.125.226.83	actions:output=2	n/a	n/a	n/a	n/a	n/a	n/a	0	0	115	600

Challenges - Floodlight

- Numerous obstacles encountered!
- Unstable releases – last stable release was in 2013!
- Outdated, incomplete documentation
- Obscure APIs, silent failures, very hard to know what we did wrong
- Had to spend 20+ hours reading its source code for debugging
- Actively communicating with Floodlight developers did help us

Challenges – Machine Learning

- Hard to choose representative input dataset
  - Research traces are too complicated
- Hard to choose good feature
- Bug in Wireshark prevents exporting packets with certain protocols
  - eg. doesn’t work for dropbox protocol “db-lsc”

Limitations

- Trace not representative & realistic:
- Only 4 kinds of flows used for training
  - in real life 100s of different flows
- Limited training size: 12000 packets
- Packets sampled from contiguous time durations

- To be improved in future work

Summary

- We use deep packet inspection and novel machine learning techniques
- Can accurately identify flows of different applications types
  - Best result 87.5% using SVM, 79% using K-Means on test sets
  - Can differentiate traffic from Skype and BitTorrent for the traffic we sampled, which Wireshark cannot tell apart.
- Can push rules with different priorities to show our control for different application traffics

Future Work

- Test on more application types
  - eg. OpenVPN, Media applications
- Try additional machine learning algorithms,
  - eg. Neural networks, Mixture of Gaussians
- Build more realistic topologies to test our framework
  - More hosts, more switches…

Thanks! Any Questions?