FPL 2010
milano, italy

Lawrence Livermore National Laboratory
USC Viterbi School of Engineering

Real-Time Classification of Multimedia Traffic using FPGA

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Weirong Jiang[1] and Maya Gokhale[2]

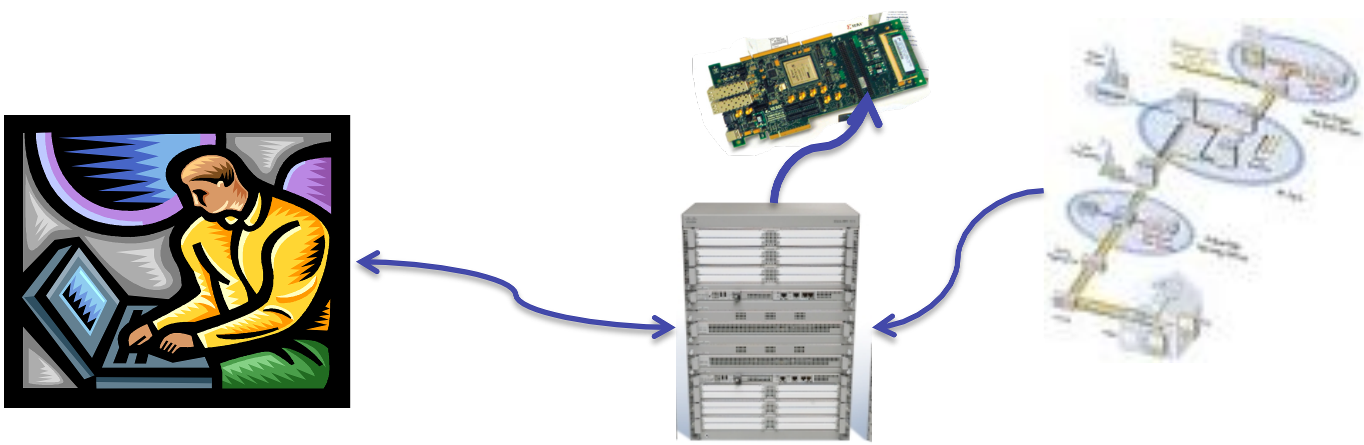
[1]University of Southern California

[2]Lawrence Livermore National Laboratory

August 31, 2010

Background

- Internet Traffic Classification
  - Identify the application type of a packet / flow
    - e.g. WWW, Email, FTP, Instant Messaging, Skype, IPTV, ...
  - Traffic Engineering, QoS, Security, …
- Multimedia Applications
  - Difficult to identify
    - Random port (P2P), encrypted payload (Skype), etc.



Existing Methods

- Port-based
  - Traditional method, e.g. WWW: 80
  - Fail for applications using random port numbers
- Deep Packet Inspection
  - Pattern matching in packet payload
  - Fail for encrypted payload
- Host Behavior
  - Connection patterns between hosts
  - Topology / Traffic –dependent
  - May not capture packets from both sides of connection
- Machine Learning
  - active research in ML approaches to traffic classification
  - few ML hardware acceleration approaches
    - C4.5 decision trees: 8Mpackets/s on NetFPGA

Statistical Traffic Classification

- Existing efforts
  - Feature selection: packet fields, flow duration
  - Machine learning algorithms
  - Focus on accuracy and robustness
- Challenge
  - Computation complexity in both training and testing
  - State size (memory requirements)
  - Achieve real-time classification

- Our goal
  - High accuracy: fraction of packets correctly classified
  - Low memory requirements
  - High classification throughput
    - Packets per second
  - Dynamic update
    - Online training
    - Intermix training and data packets

Data sets

- Data sets contain IP traffic for three categories of multimedia application
  - VoIP, IM, IPTV from http://tstat.tlc.polito.it/traces.shtm
  - Legacy applications (www, mail) from http://mawi.wide.ad.jp/mawi/samplepoint-A/2000

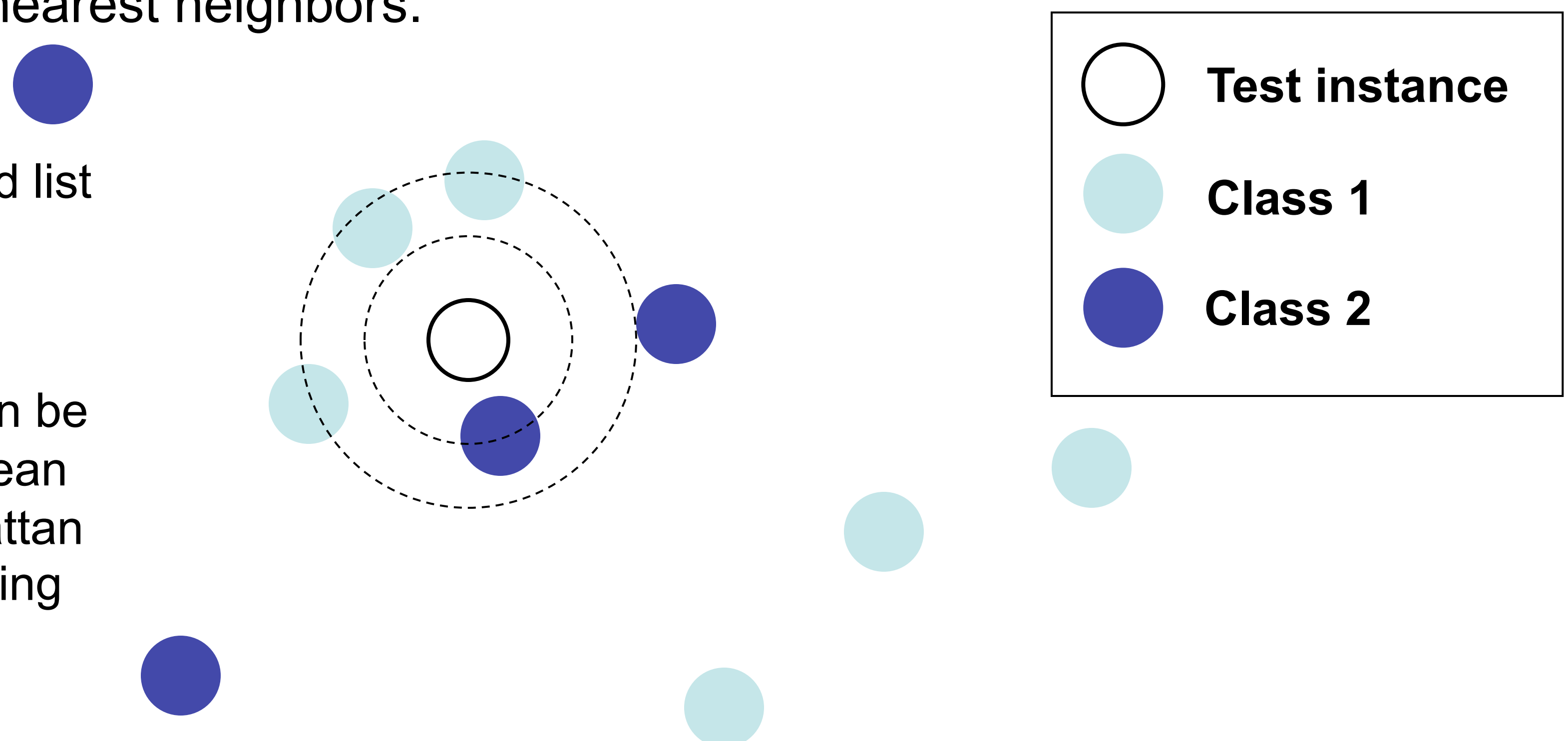
TRAFFIC TRACES

Application	Traces	Start date & time	Duration	IP protocol	# packets	Data size (MB)
Skype	Skype1	2006-05-29 02:18:41	95 hour 26 min	TCP	2357997	338.5
	Skype2	2006-05-29 02:01:25	95 hour 45 min	UDP	39627543	8396.8
	Skype3	2006-05-29 02:49:20	79 hour 3 min	UDP	3049284	231.3
Instant Messaging (IM)	MSN	2006-05-29 02:01:25	95 hour 45 min	TCP	15434573	2234.3
	YMSG	2006-05-29 02:01:26	95 hour 45 min	TCP	841221	79.1
	XMPP	2006-05-29 02:01:25	95 hour 45 min	TCP	214636	34.8
IPTV	IPTV	2008-05-06 06:19:42	5 min 32 sec	UDP	13513514	18633.8
Legacy	WIDE-2000	2000-01-01 13:59:00	1 hour 35 min	TCP & UDP	2095192	1052.5

- We use packet level features only
- IP protocol, packet size, TCP/UDP ports, TCP flags

Machine Learning Algorithm

- k Nearest Neighbors (k-NN)
  - Supervised learning
  - Assigns to a test instance the majority class type of its k nearest neighbors.



•Keep sorted list of k nearest neighbors
•Distance measure can be
  •Euclidean
  •Manhattan
  •Hamming

k-NN characteristics

- Desirable Properties
  - Proved high accuracy in traffic classification
  - Simple implementation
  - Low training cost
    - No training needed for the basic k-NN!
    - Fast update
- Challenges
  - High computation complexity (Low throughput)
    - Given **N** training samples,
    - It takes O(**N**) time to classify each test instance
    - e.g. **~ 1.6 sec** for classifying a test instance against **100K** training samples in our experiments
  - Potentially high memory requirements

K-nn algorithm

- R= training set, r in R, q is a test instance

1. Initialize a k-entry list L = {} to hold the k training samples with smallest distance to q
2. For each r in R
   1. Compute d(q,r) distance between q and r
   2. Insert r into L in sorted order; truncate L to length k
3. Assign q the label of the majority among the k training samples in L

Locality-Sensitive Hashing (LSH)

- Proposed in 1998
  - Indyk, P. and Motwani, R., "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," Proc. of 30th Symposium on Theory of Computing (STOC), 1998.
  - Approximate k-NN in high dimensional spaces
- Basic idea
  - Hash the data points using multiple hash functions;
  - for each function $h(.)$: if $d(p, q) < d(p', q')$,

    $\Pr(h(p) = h(q)) > \Pr(h(p') = h(q'))$
- Reduced computation complexity
  - # of hash tables = H
  - O(N) ➔ O(H) time for classifying each test instance

LSH Function

- Distance function: Hamming distance

Example

- p = 0101, q = 1101
- p' = 0001, q' = 1111
- Distance
  - d(p, q) = 1
  - d(p', q') = 3

- Hash function h: random single bit selection
  - $\Pr(h(p) = h(q)) = 1 - d(p, q)/4 = 3/4$
  - $\Pr(h(p') = h(q')) = 1 - d(p', q')/4 = 1/4$
- So with hamming distance and random bit selection,
  - $\Pr(h(p) = h(q)) > \Pr(h(p') = h(q'))$

K-nn with LSH algorithm

Training

1. Choose H LSH functions g_i, i= 1, … , H
2. Construct H hash tables h.
3. For each r in R
    1. For i = 1 to H
        1. $h_i[g_i(r)] = r$

Testing

1. Initialize a k-entry list L = {} to hold the k training samples with smallest distance to q
2. For i= 1 to H
    1. $r = h_i[g_i(q)]$
    2. compute d(r, q)
    3. Insert r into L in sorted order; truncate L to length k
3. Assign q the label of the majority among the k training samples in L

Performance Evaluation

- Data Set
  - Skype
  - Instant messaging (IM): MSN, Yahoo, Gtalk, ...)
  - IPTV
  - Legacy applications: WWW, Email, DNS, ...
- Features
  - IP protocol (8 bits)
  - packet size (16 bits)
  - TCP/UDP ports (16 bits)
  - TCP flags (8 bits)
- Performance metrics
  - Accuracy
    - # of correctly classified packets / # of all packets
  - Throughput
    - # of packets per second (PPS)

Software Implementation

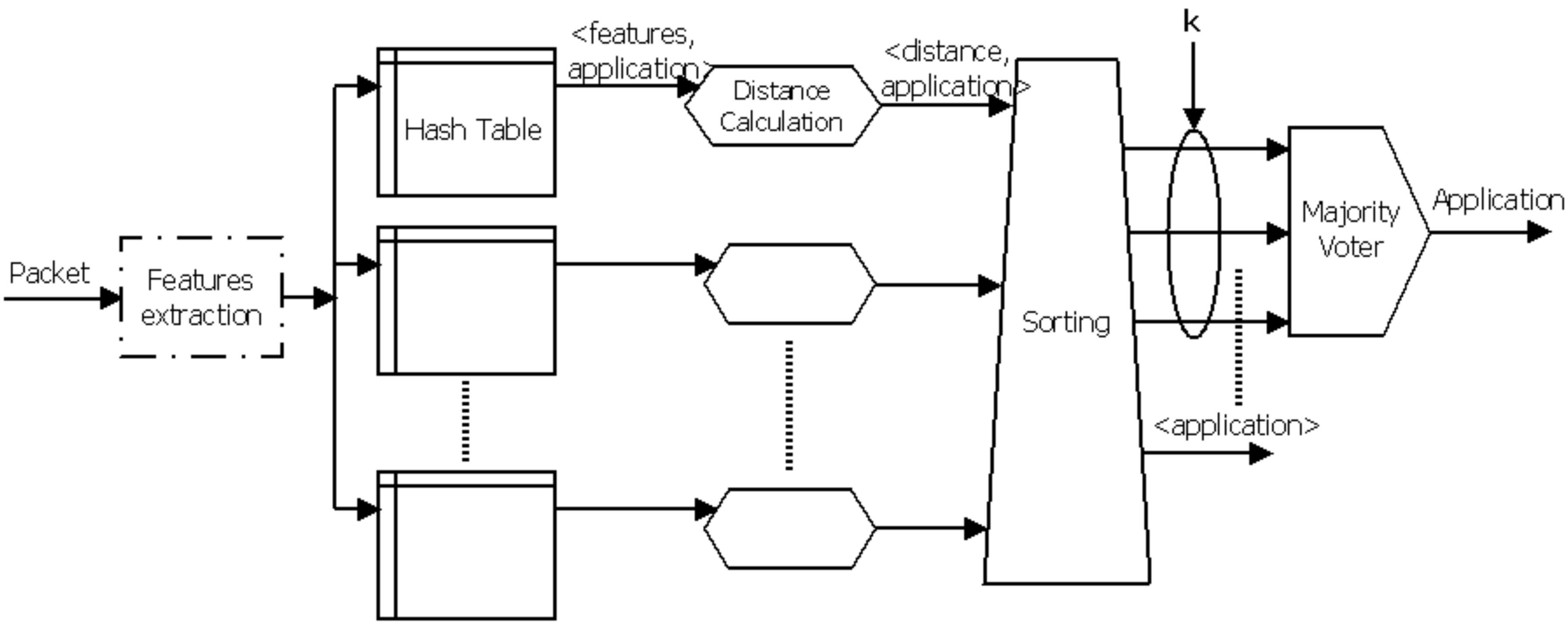
# of Training samples	Classification accuracy		Classification time per test instance	
	k-NN	LSH	k-NN	LSH
100	85.48%	81.24%	2.11 msec	0.023625 msec
1K	87.23%	87.24%	15.15 msec	0.024625 msec
10K	99.90%	99.79%	164.04 msec	0.025500 msec
100K	**100%**	**99.97%**	**1568.11 msec**	**0.025750 msec**

- Corresponding maximum throughput: 40 KPPS

Hardware Accelerator

- Parallel & pipelined architecture
- Pipelined bitonic sorting network

- Parameterized architecture
  - k, # hash tables H, hash table size M



Number of hash tables

- More hash tables gives better accuracy and correspondingly larger resource utilization

IMPACT OF THE NUMBER OF HASH TABLES

# of Hash tables	Overall accuracy	Slice usage	BRAM usage
2	73.58%	1%	3%
4	97.56%	5%	9%
8	99.56%	13%	20%
12	99.83%	22%	30%
16	99.97%	27%	40%

Hash table size

- Larger hash table increases accuracy
- No impact on BRAM usage until 2K due to fixed BRAM size allocations

IMPACT OF HASH TABLE SIZE

Hash table size	Overall accuracy	Slice usage	BRAM usage
256	99.34%	28%	40%
512	99.83%	29%	40%
1K	99.97%	27%	40%
2K	99.98%	29%	80%

FPGA Implementation

- Parameters used: k=2, H=16, M=1K
- Xilinx Virtex 5 xc5vlx50t with -1 speed grade on a Xilinx ML555 development board

	Available	Usage
# of Slices	7200	27%
# of IOs	480	25%
# of BRAMs	60	43%
Total memory	2160 Kb	40%
Clock rate	125 MHz	

- Throughput (w/ dual-port RAM):
  - Classification: 250 MPPS
    - **80 Gbps** for minimum size (40 bytes) packets
  - Training (Update): 125 MPPS = 40Gbps

Performance Summary

- Comparison

	k-NN (SW)	LSH (SW)	LSH (FPGA)
Accuracy	100 %	99.97%	99.97%
Throughput	0.6 PPS	40K PPS	250M PPS
Speedup	1	$6.7\times\mathbf{10^4}$	$4.2\times\mathbf{10^8}$

- PPS: # of packets per second
- Latency improvement
  - $6\times\mathbf{10^4}$
  - $1.5\times\mathbf{10^7}$

Concluding Remarks

- Statistical traffic classification is promising for classifying multimedia traffic
  - Attracted a lot of research interests recently
- Real-time performance demanded (Yet little work has been done!)
  - Algorithm-level optimization
  - Hardware accelerators
- Our contributions
  - Propose using LSH for high-speed traffic classification
  - First 10+ Gbps FPGA design for traffic classification
  - Achieving high accuracy and high throughput
- Open Problems
  - Other machine learning algorithms
  - Other packet/flow/network -level features
  - Other hardware accelerators (multi-core, many-core, ...)

Q & A

- Thanks