中原大學 Chung Yuan Christian University

Implementing On-line Sketch-Based Change Detection on a NetFPGA Platform

Y.K. Lai, N.C. Wang, T.Y. Chou,
C.C. Lee, T. Wellem, H.T. Nugroho

1st Asia NetFPGA Develover Workshop, June 13-15, 2010, KAIST, Daejeon, South Korea

Introduction

- **This project implements a Sketch-based change detection on network traffic on NetFPGA**

- **The change detection scheme is based on the scheme proposed by Krishnamurthy *et al*. [1]**
  - Software implementation
  - Uses k-ary sketch

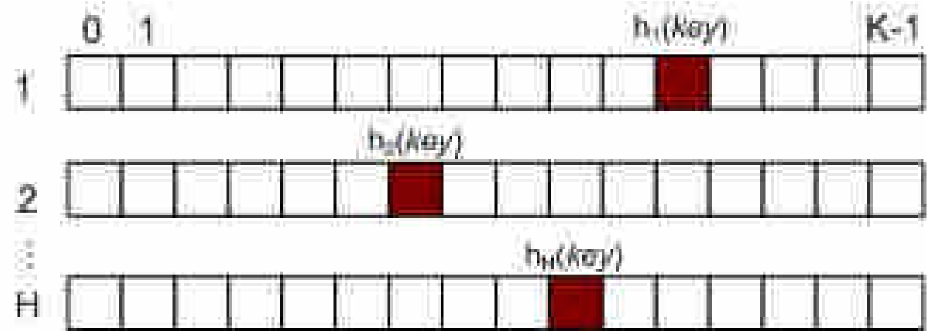
[1]. B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: methods, evaluation, and applications," Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, Miami Beach, FL, USA: ACM, 2003, pp. 234-247.

Chung Yuan Christian University

Sketch

- **Data structure to build summary of data stream**
  - Space-efficient
  - Accuracy with probabilistic guarantee

- **K-ary sketch**
  - Array of counters *C[i][j]* (i=1…H, j=0…K-1)
  - Indexed by 4-Universal hash functions, $h_1 \ldots h_H$



Sketch-based Change Detection

- **Sketch module**
  - Summarizes traffic using sketch for each time interval, *t*
  - *Observed Sketch*, $S_o(t)$
- **Forecast module**
  - Using the observed sketches from past intervals, it uses a *forecasting model* to build *Forecast Sketch*, $S_f(t)$
    - Forecast model: Moving Average (MA), EWMA, ...
  - Computes the *Forecast Error Sketch*, $S_e(t)$.
    - $S_e(t) = S_o(t) - S_f(t)$

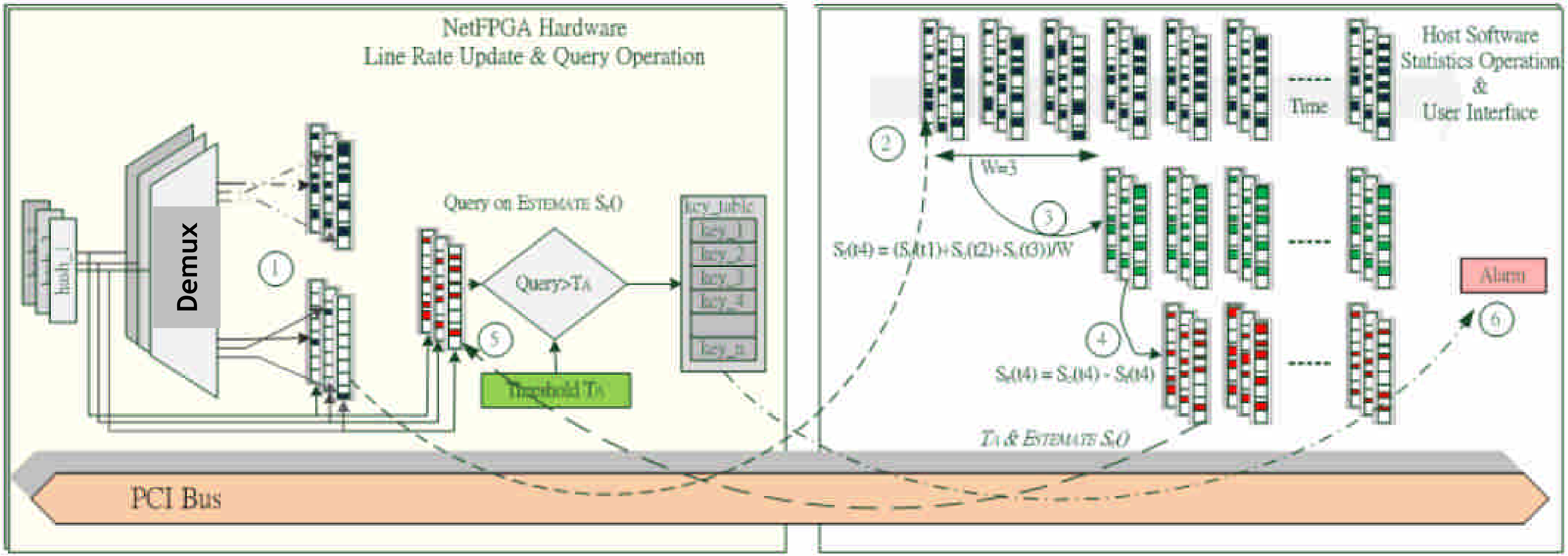
Sketch-based Change Detection

- **Change detection module**
  - Computes alarm threshold, T_A based estimated 2nd moment of $S_e(t)$ and parameter T determined by application

$$T_A = T.\left[F_2^{estimate}(S_e(t))\right]^{\frac{1}{2}}$$

  - $S_e(t)$ is used to determine significant changes
  - For any *key a*, the *estimated forecast error* is ESTIMATE($S_e(t)$, *a*)
  - Flows with *estimated forecast error* greater than T_A will be reported

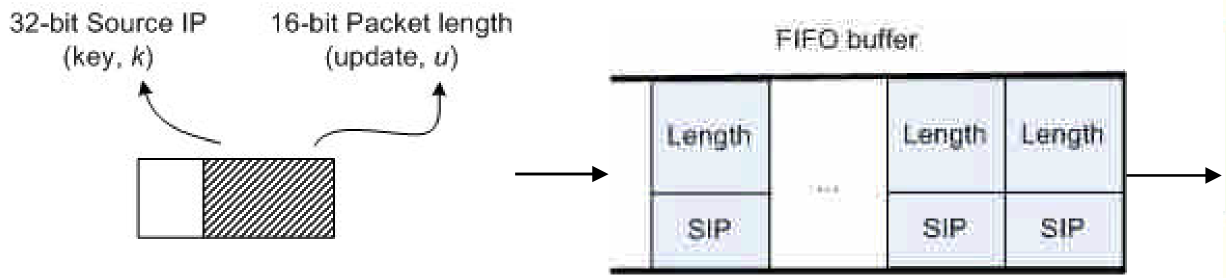
System Architecture

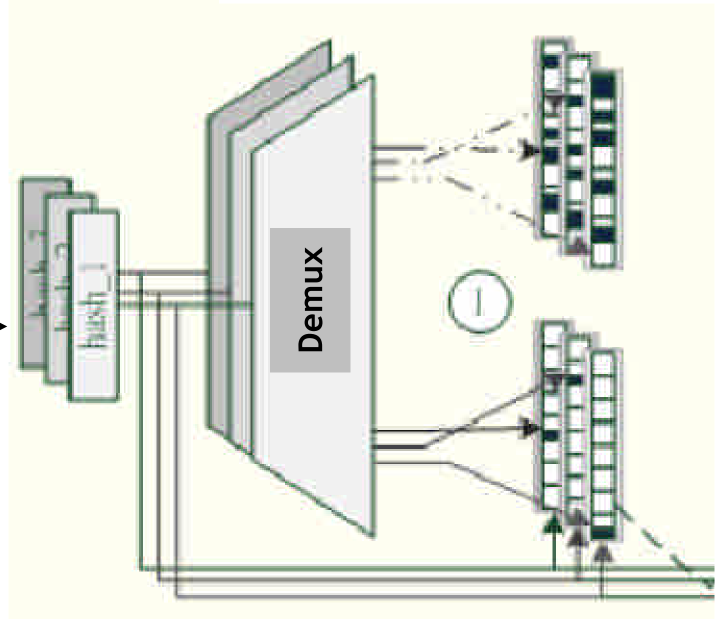


Hardware Components

- **Sketch Module**
  - Sketch update process
    - 2 sketches in SRAM

Observed Sketch, So(t)

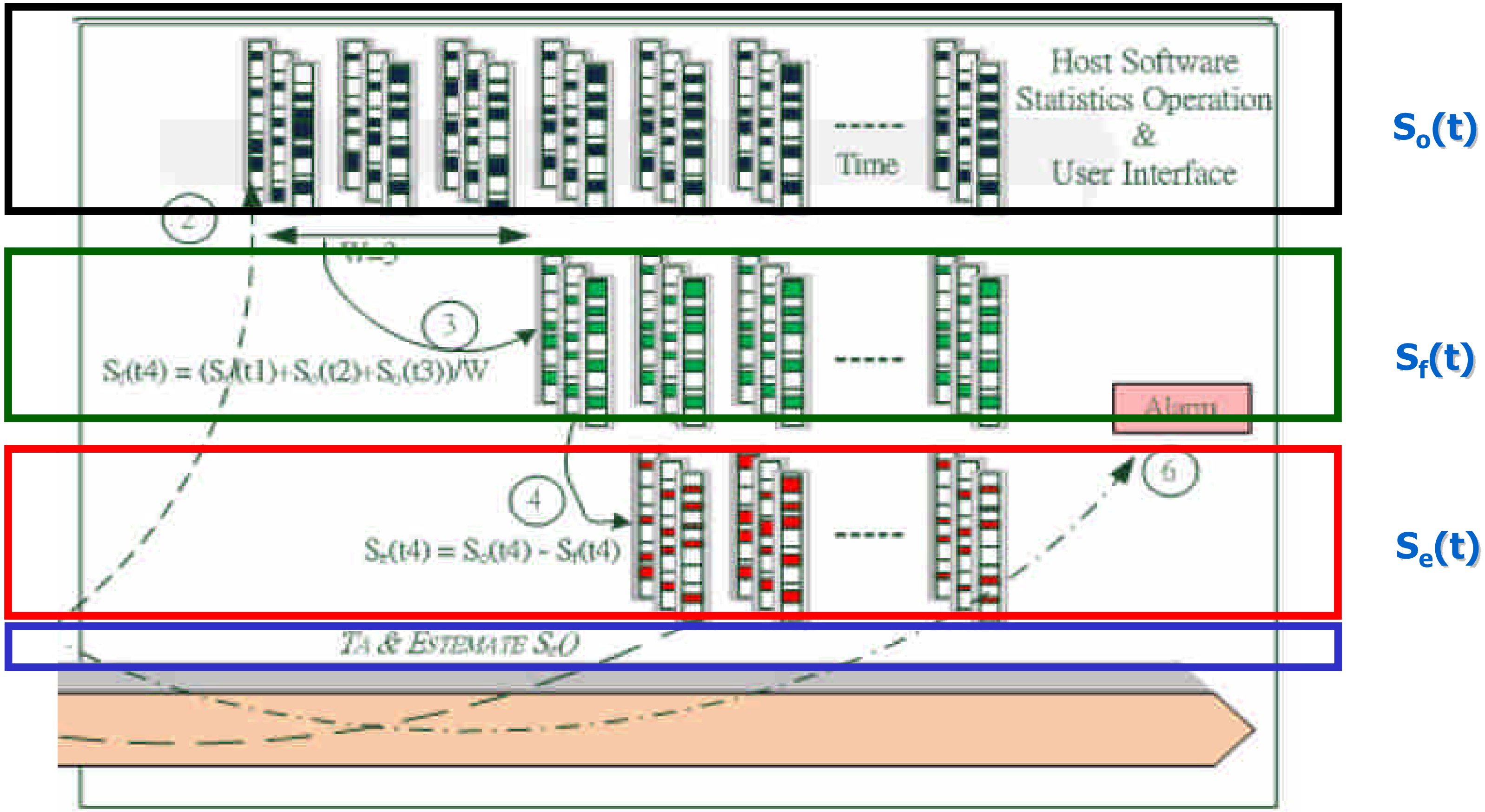




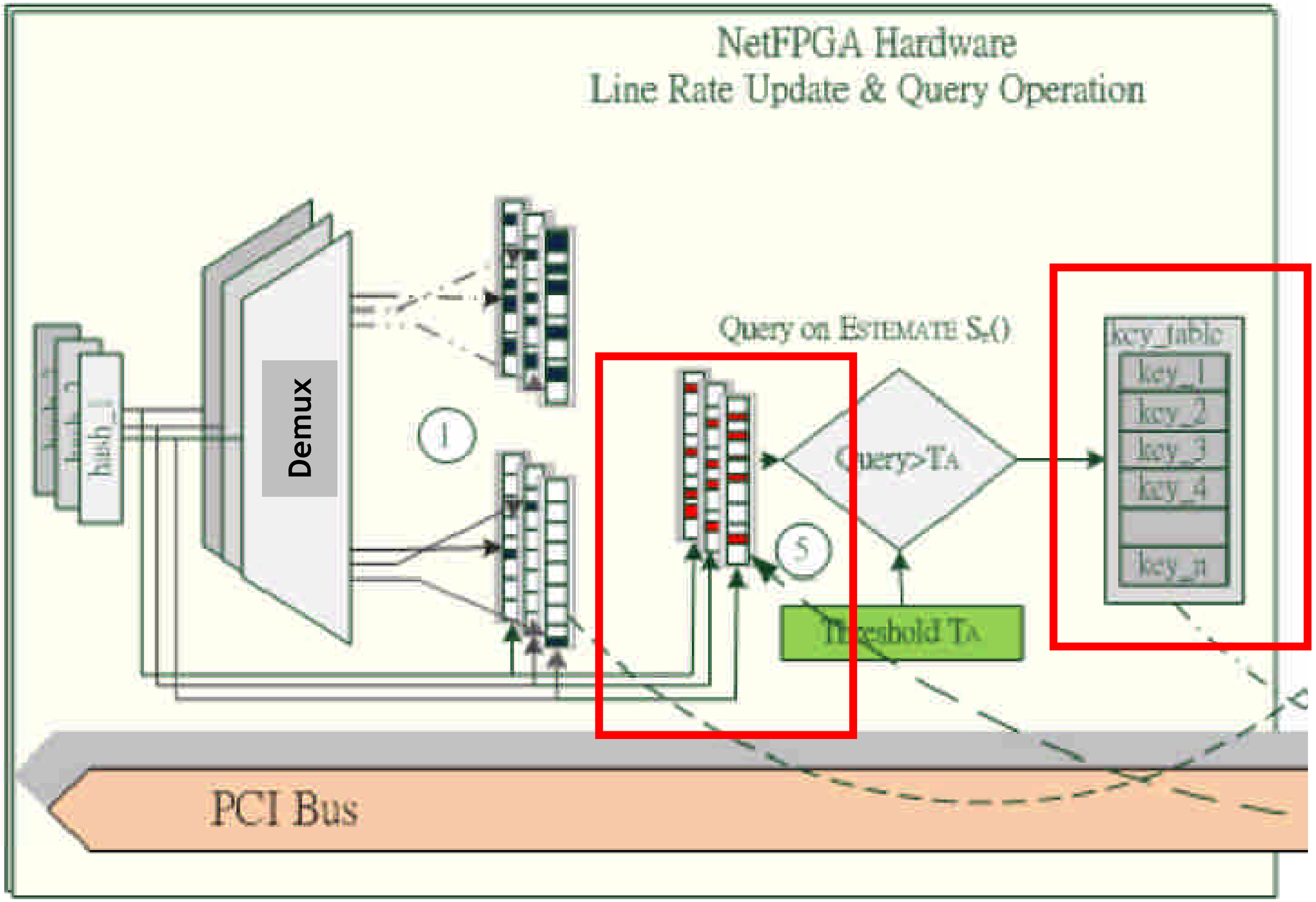
中原大學
Chung Yuan Christian University

Software Components

- Estimator

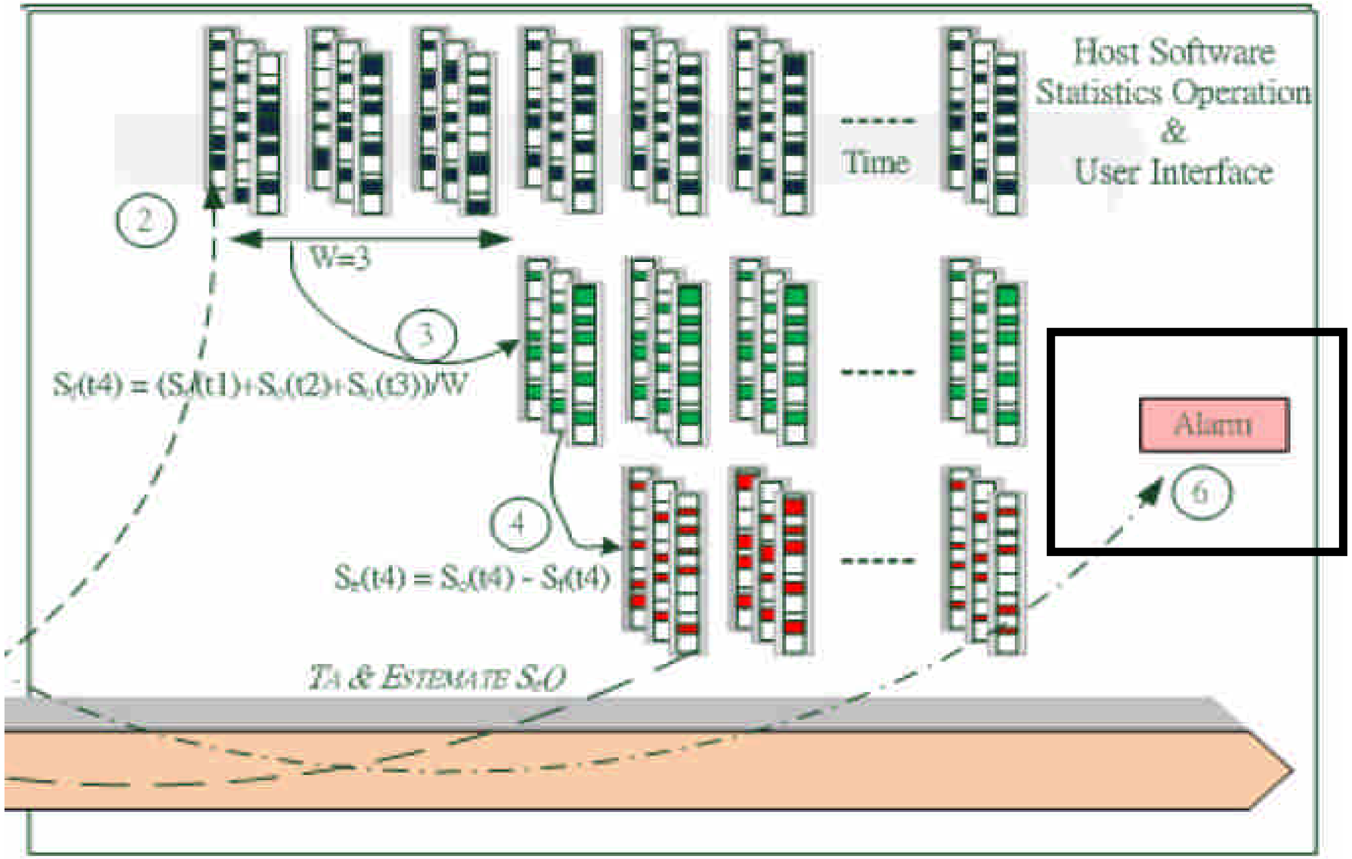


Software Components



Software Components

- Estimator



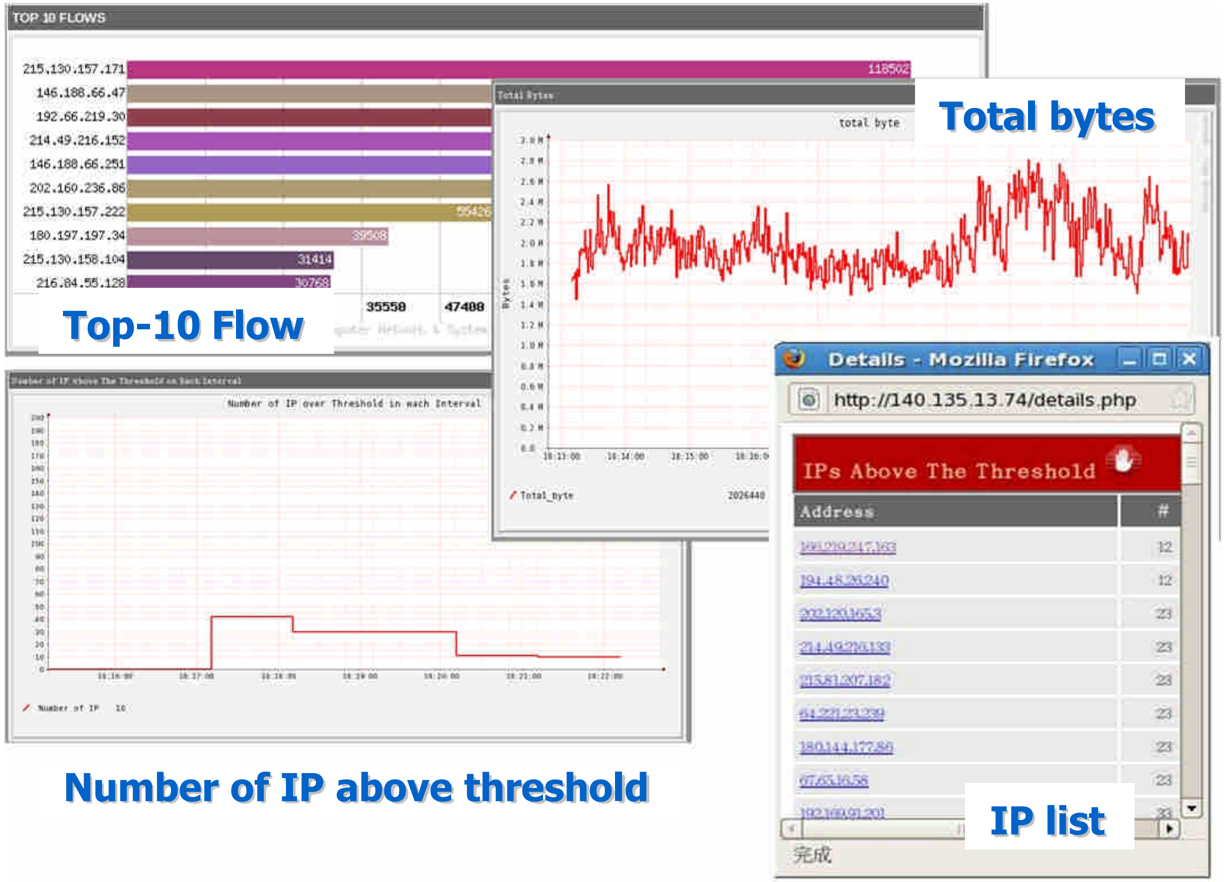
Software Components

- **How to get the key to query $S_e(t)$?**
  - We use the keys after $S_e(t)$ has been constructed
    - Use current incoming key to query previous forecast error sketch
  - Advantages
    - Avoid the need of two-pass ("touch" the stream twice)
    - Avoid the need to store all the keys
  - Drawback
    - Miss the keys that do not appear again after they experience large change

中原大學
Chung Yuan Christian University

Software Components

- Graphical User Interface (GUI)



1st Asia NetFPGA Develover Workshop, June 13-15, 2010, KAIST, Daejeon, South Korea

Evaluation

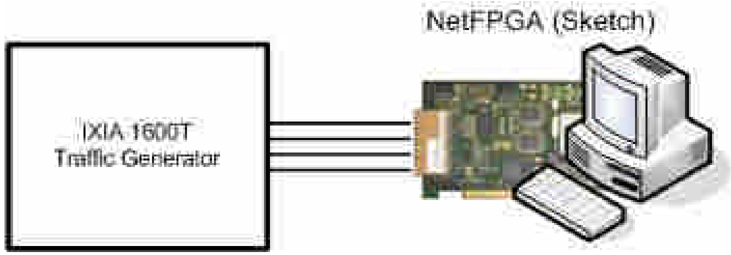
- **Trace-driven experiment**
  - MAWI trace files

Trace file	Duration	Number of distinct flows (based on SIP)
200302270000.dump	15 min.	51,788
200304022100.dump	15 min.	286,369

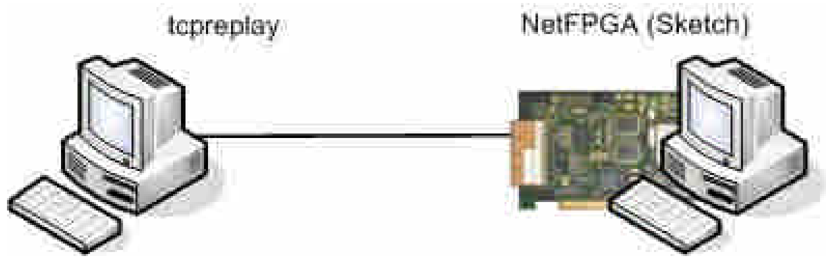
- **Parameters**
  - H=3, K=32K Window Size (W) = 3, Interval = 60 seconds
- **Hash function**
  - 4-Universal, pipelined multiplier

Evaluation

- **Testing Topology**
  - Sketch update testing



  - Accuracy testing



Evaluation

- **Sketch update testing**
  - Metric: Percentage of packet loss
- **Accuracy**
  - Metric: False negative and false positive rates
- **Resource Utilization**
  - Metric: Percentage of resources used

Results

- **Sketch update**
  - Can achieve line-rate update
  - 0.16% packet loss under stress test using 4 Gbps minimum-sized frame
- **Accuracy**
  - The system can successfully detect flows (source IPs) whose change is above threshold

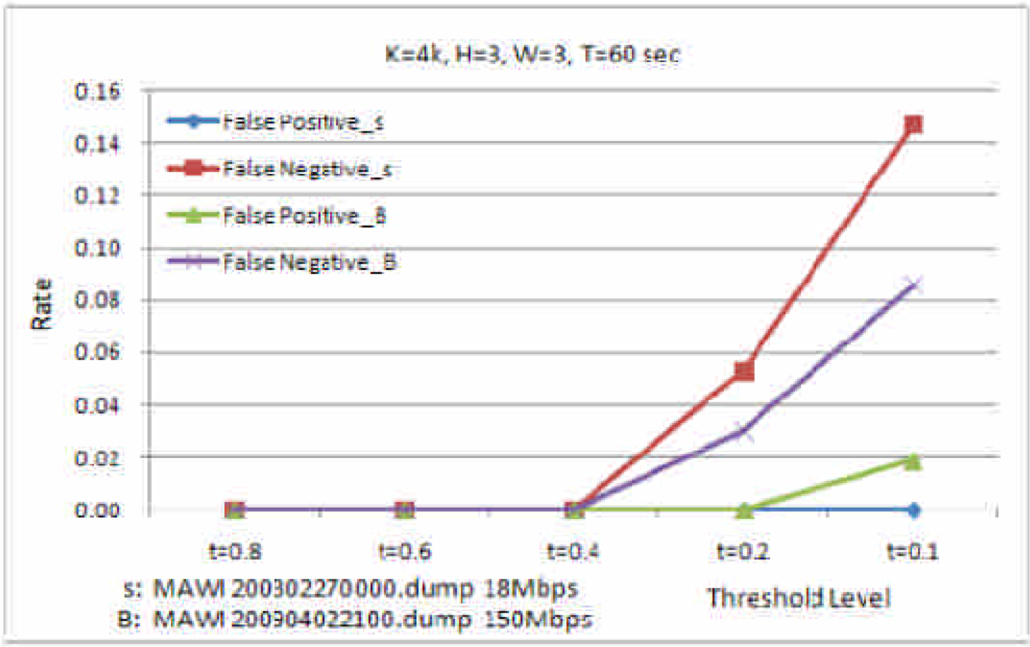
Accuracy with various threshold parameter T

T	0.8	0.6	0.4	0.2	0.1	0.05	0.02
False Positive	0	0	0	0	0	0.005	0.008
False Negative	0	0	0	0.05	0.11	0.14	0.17

中原大學
Chung Yuan Christian University

Results

- **Software simulation**
  - H=3, K=4K



Results

- **Resource Utilization**
  - 4-Universal hash (pipelined multiplier)

Resources	Utilization	Percentage
Slice Registers	22,687 out of 47,232	48%
4 input LUTs	20,433 out of 47,232	43%
Occupied Slices	16,671 out of 23,616	70%
RAMB16s	144 out of 232	62%
MULT18X18s	72 out of 232	31%
IOBs	356 out of 692	51%

Conclusion and Future Work

- **We have implemented a network traffic change detection system on NetFPGA**
  - Can achieve online, one-pass change detection

- **Further improvements**
  - Integration with router or switch design
  - Network-wide anomaly detection system

Thank you