FERMILAB-SLIDES-18-122-CD

GoldenEye: stream-based network packet inspection using GPUs

Qian Gong, Wenji Wu, Phil DeMar

The 43nd IEEE Conference on Local Computer Networks

October 4, 2018

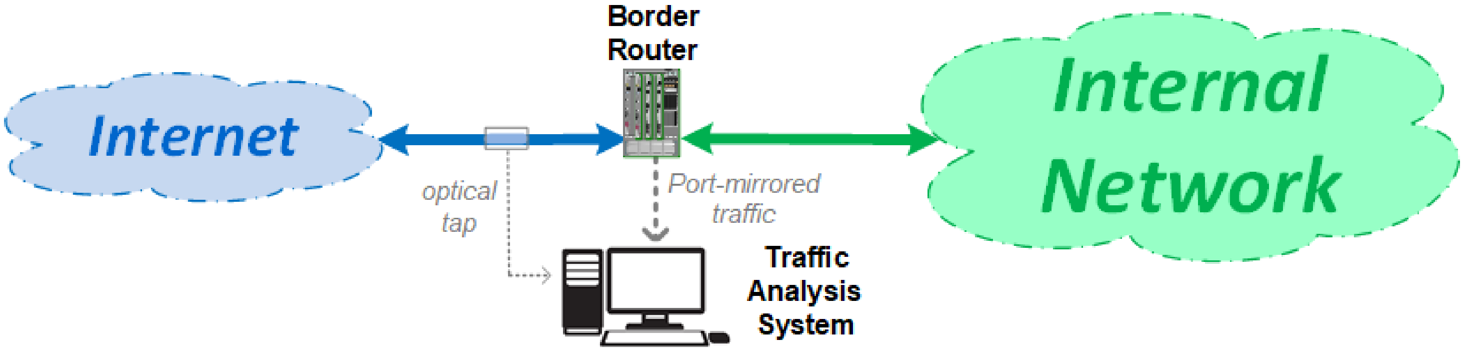
This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

Outline

- Motivation of GPU-based traffic analysis
- Framework of GPU-based traffic analysis
- Performance evaluation
- Conclusion & future work

Network Traffic Analysis

- Network traffic analysis tools provides indispensable information for
  - Operation & management
  - Performance troubleshooting
  - Network security
  - Statistical purpose



- Basic functions:
  - Profile traffic activities
  - Scan traffic content for suspicious patterns signatures

Task Overview

Stateful packet processing

- Track and maintain the states of network functions:
    - TCP connections
    - Sub-string matches in intrusion detection systems

Timely response

- Fast and reliable network data processing at a link speed

Protect traffic integrity

- Packet shouldn’t be lost in processing cycle

Challenges in data and state management

Data Management Challenges and Solutions

Challenges:

- High-speed networks
  - 10/25/40GE-connected serves
  - 100GE backbone technologies are commonplace

Millions of packets generated & transmitted per second

- Complex packet analysis algorithms
  - Algorithms are increasingly complex as security threats become more sophisticated
  - Need a flexible and programmable computing platform

Data Management Solutions

Solutions:

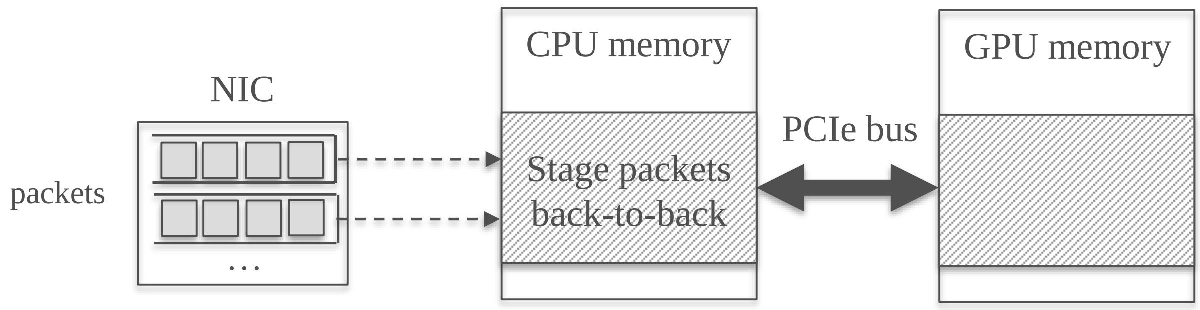
- Heterogeneous data management
- GPU-centric computing
  - GPU is specialized for data-parallel, large-throughput computations
  - Thousands of cores for massively parallelism
  - Tolerance of memory latency

Features	NPU/ASIC	CPU	GPU
High compute power	Varies	✗	✓
High memory bandwidth	Varies	✗	✓
Easy programmability	✗	✓	✓
Data-parallel execution model	✗	✓	✓

Packet Processing on Heterogeneous Architectures

Data processing flow:

- CPU receives packets from NIC, parses headers and *batches* them in an input buffer.
- When a specified batch size or a preset time limit is reached, the input buffer is transferred to the GPU memory via PCIe.
- A set of GPU kernels are then launched to perform tasks such as IP address matching, cryptographic operations, and deep packet inspection.
- The results are transferred back to the CPU memory to guide further actions.



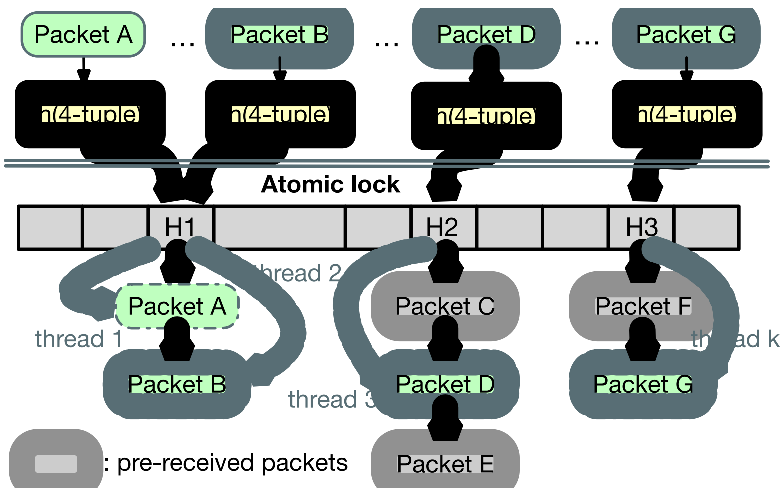
Packet batching can be a feasible way to improve GPU utilization, but it increases difficulties in stateful packet processing…

State Management Challenges

Challenge 1: flow management & stream reassembly

- Stateful network functions must both track the states of network connections and scan network packets at a per-flow level.
- Flow state management and stream-reassembly require state synchronization when dealing with packets from the same connection.
- Limited data parallelism when less simultaneous TCP connections are present.

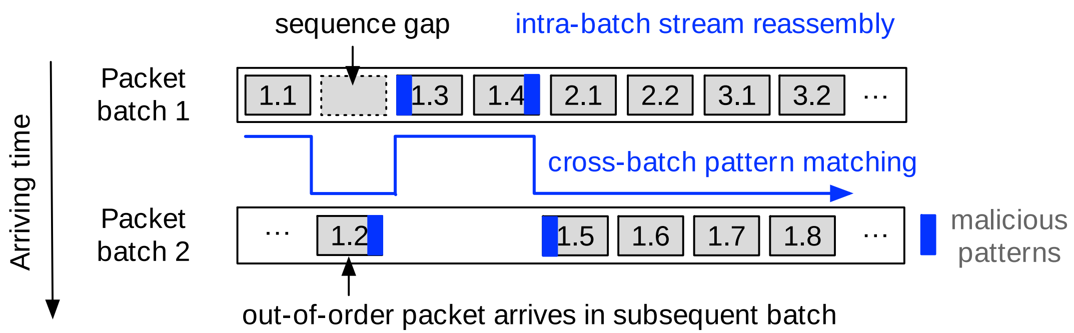
Conventional hash-based approach requires atomic locks with packets from the same TCP flow connection, and is prone to ambiguity caused by hash-key collision.



State Management Challenges

Challenge 2: Inter-batch state connection

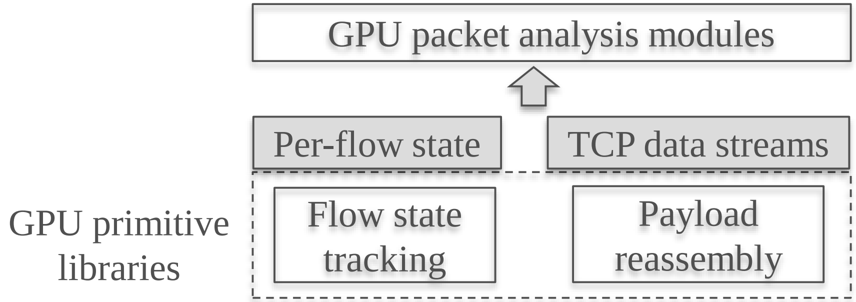
- Stateful packet inspection must detect signatures that straddle packet boundaries.
- GPU's batch-processing mechanism requires maintaining connection states and tracking potential sub-matches across input batches.



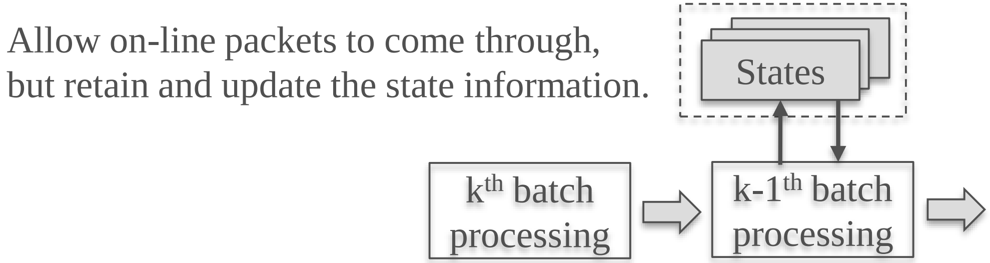
Stateful packet inspection must detect and memorize the sub-matches across input batches.

State Management Solutions

- Parallel flow management and stream processing via GPU sort and prefix-scan
  - Sort and prefix-scan are extremely fast on GPU (over ten billions of elements/sec).



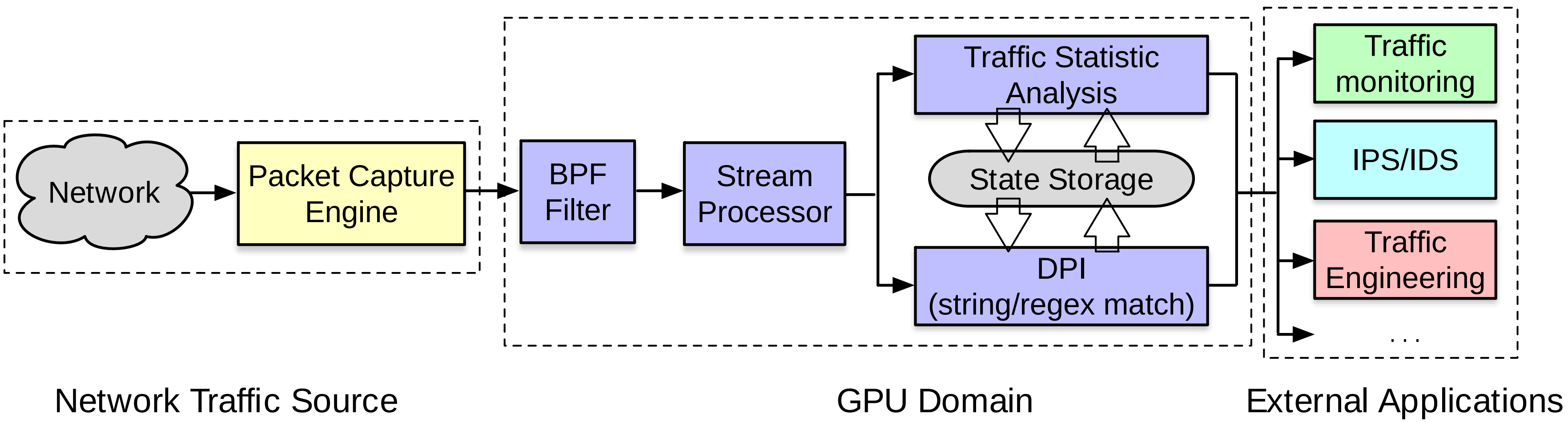
- Inter-batch network function state connection
  - Developed a buffer-free, cross-packet/batch pattern matching algorithm.
  - Combine the state and context information with packets in subsequent batches.



GoldenEye Network Traffic Analysis Framework

GoldenEye Modules

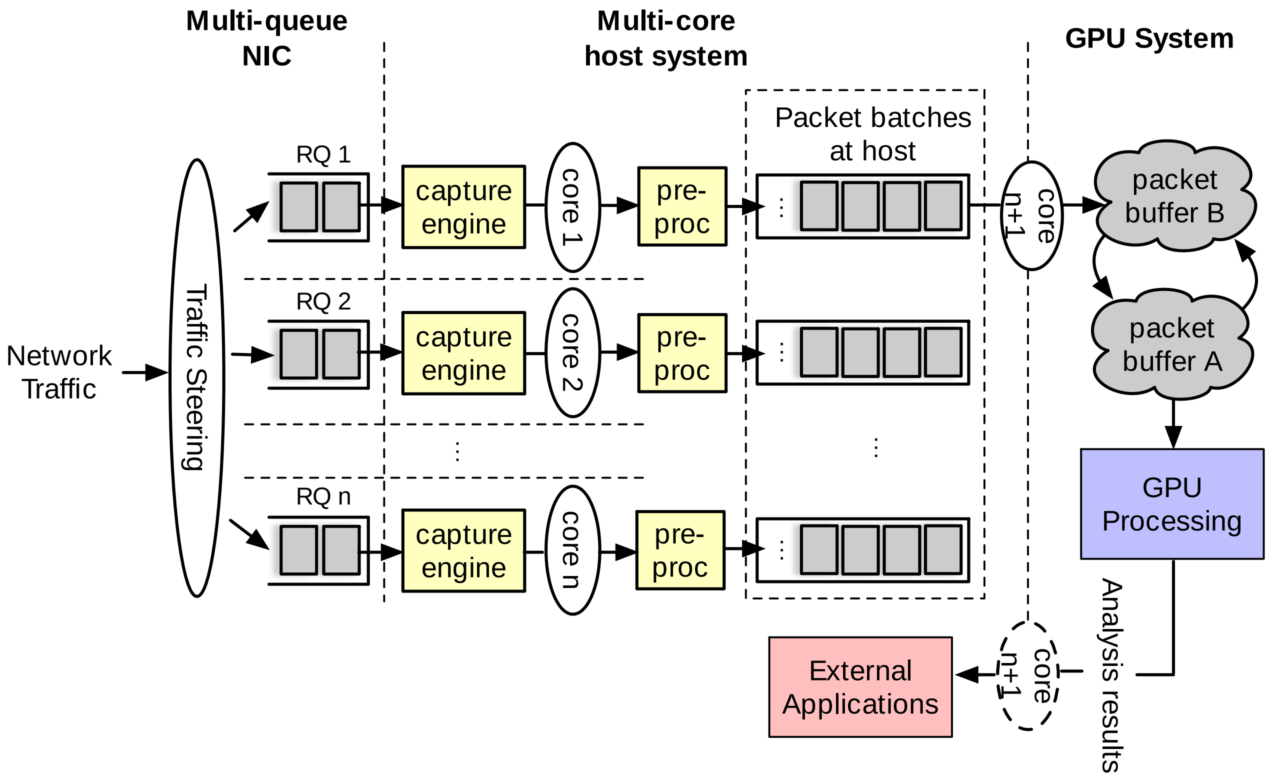
- Packet capture & pre-processing
- BPF filter
- Stream processor
- Traffic statistic summary
- Deep packet inspection
- State buffer



Packet Capture and Processing on Multicore Systems

Logics:

- Multithreading packet capturing and pre-processing
- Queue packets for batch processing
- Dual-buffer for concurrent data transfer and GPU computing



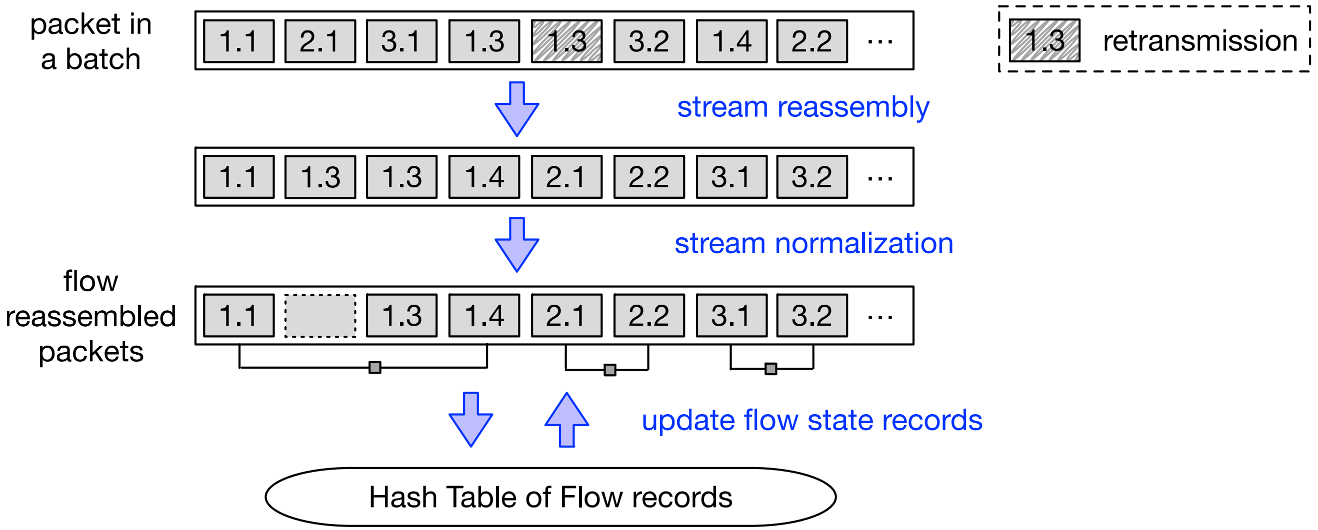
GPU-centric Stream Processor

Tasks:

- Monitor the states of TCP connections.
- Reassemble TCP packets into bi-directional byte-streams.

Implementations:

- Stream reassembly: sort packets into streams by their TCP 4-tuples and sequences.
- Flow state tracking: compare the stream states against existing connections.
- Stream normalization: rescan flow-reassembled packets and remove retransmission.



Traffic Statistical Analysis

<u>Main strategy</u>:

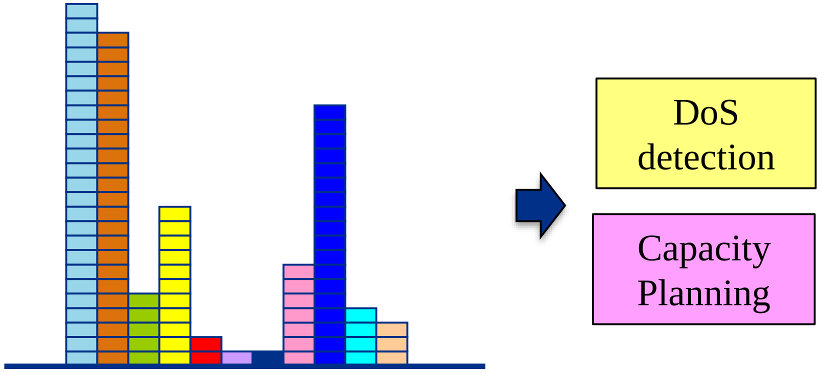
- Similar to the TCP flow management function, GoldenEye's statistical aggregation module is built with a set of primitive GPU sort and prefix-scan operations.

<u>Example use cases:</u>

- Host traffic monitoring

Src IP	Src Port	Dest IP	Dest Port	Proto	Pkt Sent	Byte Sent
131.2.3.0	80	10.1.2.4	998	TCP	32	16484
10.1.2.4	998	131.2.3.0	80	TCP	121	179841

- Heavy-hitter Detection



Stream-based Deep Packet Inspection

Tasks:

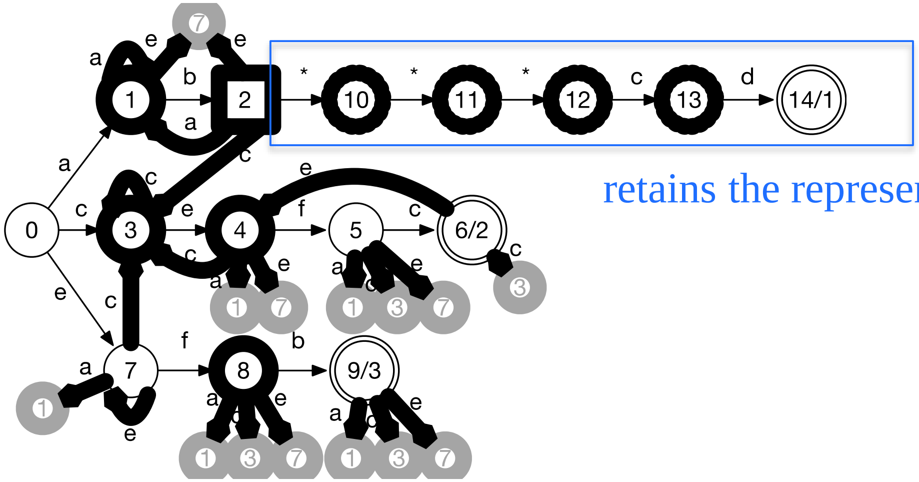
- Intra-batch pattern matching:
  - Perform pattern matching over stream-reassembled packets in the same batches.

- Inter-batch pattern matching:
  - Detect and reconstruct signature patterns that straddle batch boundaries.

Intra-batch Signature Matching

General pattern matching algorithm: hybrid-FA[1]

- HFA compresses the states of DFA by keeping any subset whose expansion would cause state explosion in an NFA form.



retains the representation in an NFA form

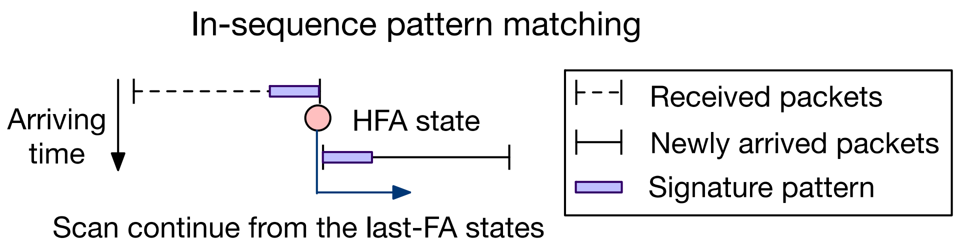
Implementation of intra-batch pattern matching:

- Each GPU thread takes one packet.
- Perform cross-packet detection at a per-flow basis.

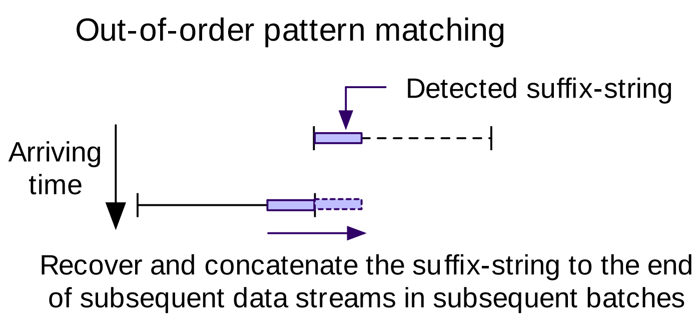
[1] Michela Becchi, Patrick Crowley, “A hybrid finite automaton for practical deep packet inspection”, 2007 ACM CoNEXT conference.

Inter-batch Signature Matching

- In-sequence pattern matching
  - Matching process of subsequent stream fragments will continue from the last FA-states of the previous fragments.



- Out-of-order pattern matching
  - Look for regex-suffixes in out-of-order streams.
  - Recover the string of previous matches and concatenate it to stream fragments that arrive latter to fill the hole.



Out-of-order Pattern Matching

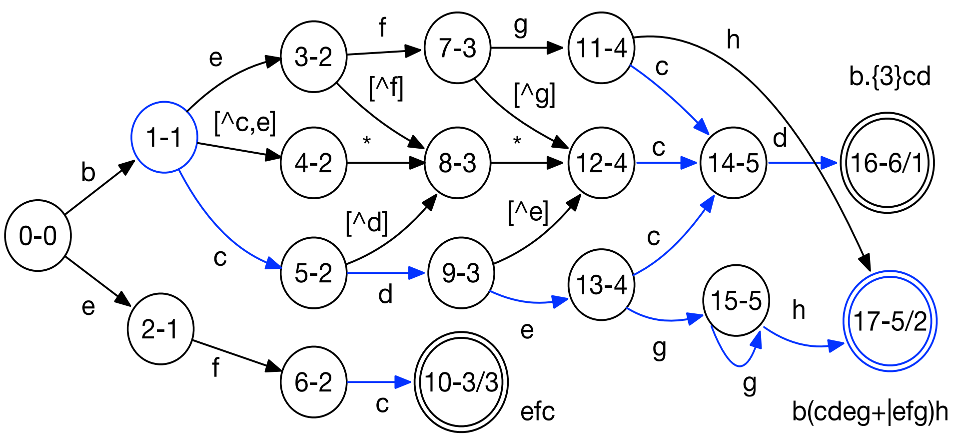
Suffix-regex detection: search streams with all potential initial states

- Parallel among both out-of-order stream fragments and the possible initial states of a search

Suffix-string reconstruction:

- Retain the first and the last FA-states of any partial-matches
- Reconstruct sub-regexes by relating states to their “depths” in the original regexes

Example: search suffix-regexes in the stream of “cdegggh”



Optimization w/ Chained Expression

Logics:

- Speed up DPI with string-based filters
- Reduce memory consumptions by breaking complex regexes into chained pieces

Strategy:

- Convert a regex into one of the three forms: <str><regex>, <regex><str>, or <regex><str><regex>.
- Process packet streams first through the string-filter.
- The regex engine will not be triggered until the string guarded by its side happens.

Performance Evaluation

Testbed systems:

- Dual Intel E5-2650 v4 CPU (12 cores per socket)
- NVIDIA K40 GPU

Traffic trace:

- Traffic source
  - Intrusion detection dataset created by Canadian Institute of Cybersecurity (CICIDS)
  - Science data flow mirrored from the border router at Fermilab (Fermilab)

- Traffic pattern

	Trace size	# of packets	# of TCP connections	Mean packet-size
Fermilab	9.8 GB	8.7×10^6	54.78×10^3	1118-byte
CICIDS	17.1 GB	44.34×10^6	1.10×10^6	386-byte

Regex datasets:

- 104 and 192 spyware and malware signatures snapshot from Snort 2.9.7.2

Performance Evaluation

Memory footprint

- Stream reassembly

Traffic Trace	Default	
	Buffer & reassembly	GoldenEye
Fermilab	307.81 MB	44.20 MB
CICIDS	186.32 MB	48.05 MB

- Chained regex FAs

Regex	Chained Regex			DFA		HFA	
	string-filter	regex engine					
malware	25.62 MB	5510 states	5.46 MB	exploded	N/A	~2×10^6 states	~2 GB
spyware	26.64 MB	3637 states	3.74 MB	~7×10^6 states	~7 GB	~1×10^6 states	~1 GB

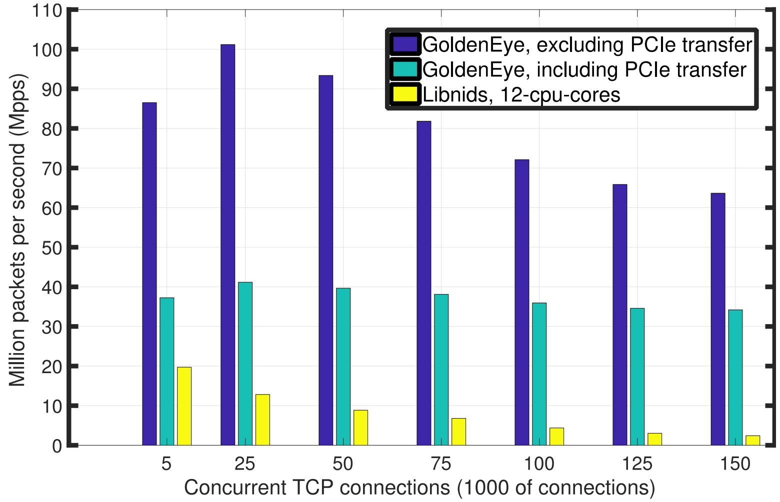
Performance Evaluation

Flow tracking & TCP reassembly

- Performance with real traffic

	Fermilab	CICIDS
GoldenEye wo/ PCIe transfer	623.30 Gbit/s	335.73 Gbit/s
GoldenEye w/ PCIe transfer	552.30 Gbit/s	232.487 Gbit/s
Libnids[1] (12 CPU cores)	186.65 Gbit/s	31.102 Gbit/s

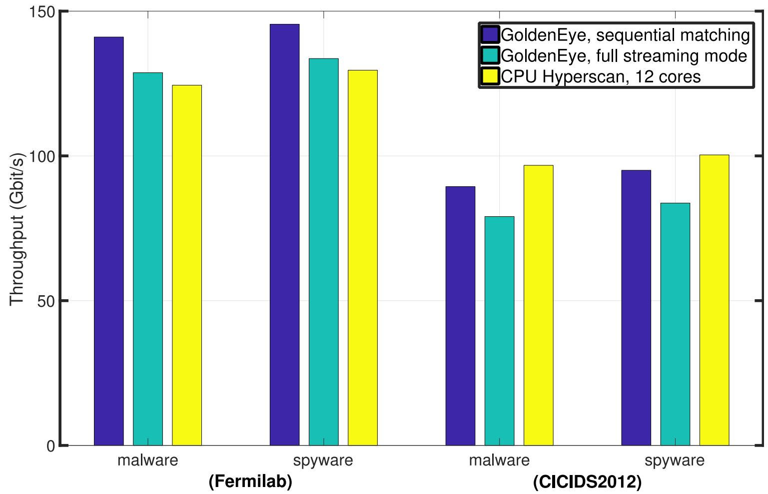
- Scalability to the number of concurrent TCP connections



[1] R. Wojtczuk, “Libnids. http://libnids. sourceforge. net.”

Performance Evaluation

Stream-based regex matching



Consolidate DPI application

Regex	Fermilab	CICIDS
malware	105.55 Gbit/s	62.13 Gbit/s
spyware	108.69 Gbit/s	64.95 Gbit/s

Hyperscan: “a high-performance multiple regex matching library” https://01.org/hyperscan

Conclusion & Future Works

GoldenEye:

- Provides a fast, memory efficient packet processing framework for GPU platforms, capable of statistical and stream-based payload analysis.
- Reassemble TCP streams in GPU and match signature patterns across packets, without requiring system to buffer and rescan packets or limit scanning to a fixed window of historical data.

Future Directions:

- Continue to add new features to support ever complex network tasks.
- Combine our packet processing functions with advanced learning algorithms to build behavior-based network automate detection.