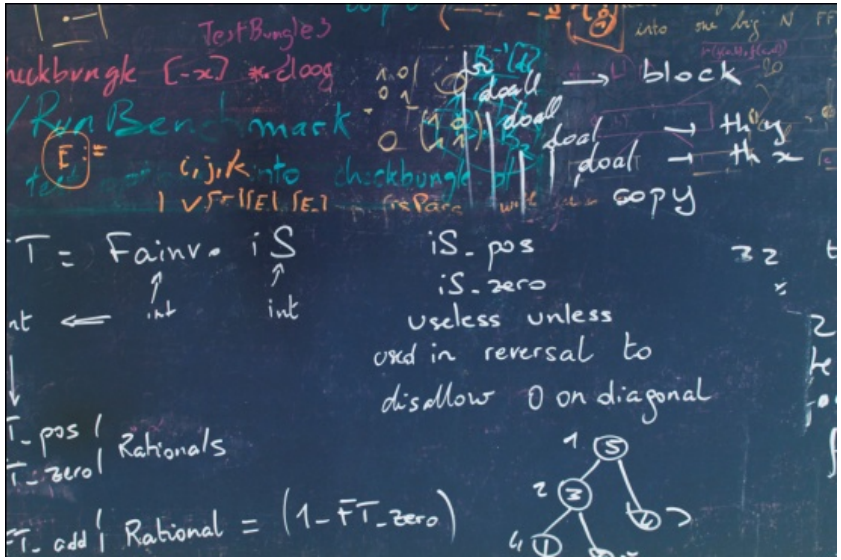
Scalable Cyber-Security for Terabit Cloud Computing

2012 IEEE High Performance Extreme Computing

Jordi Ros-Giralt / giralt@reservoir.com



Reservoir Labs

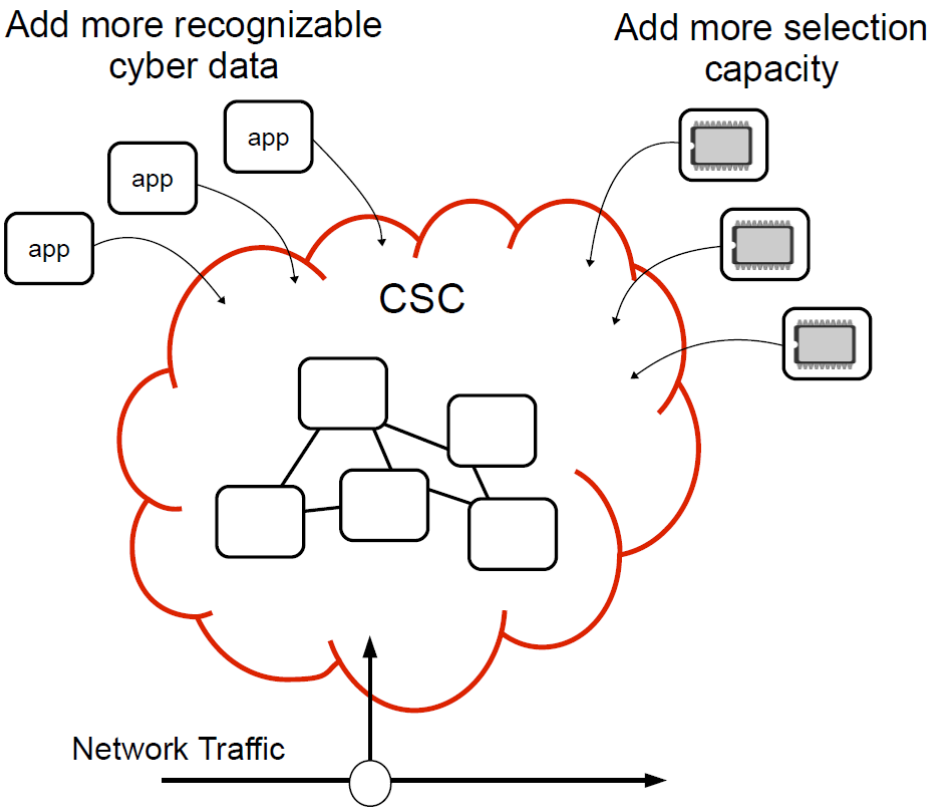
Objectives

- To be able to do packet analysis at very high-speed rates.
- To be able to intelligently move packets from node to node at very high-speed rates.

Two fundamental problems: packet analysis and packet forwarding

Feature Needed (at very high speed rates):	Application:
Packet analysis →	Cyber security
Packet forwarding →	Cloud computing / load balancing

Cyber Security Cloud: Definition

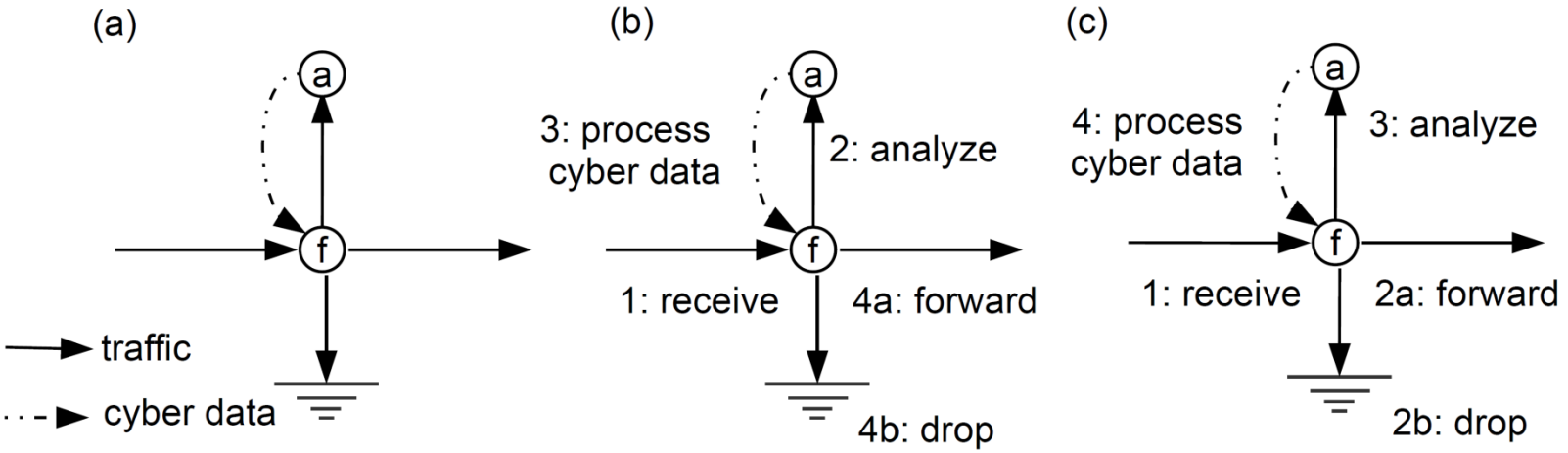


→ PAAS model – the consumer has control over the deployed cyber-security applications and their configuration and it provides support for any of the four deployment models (private cloud, community cloud, public cloud, hybrid cloud)

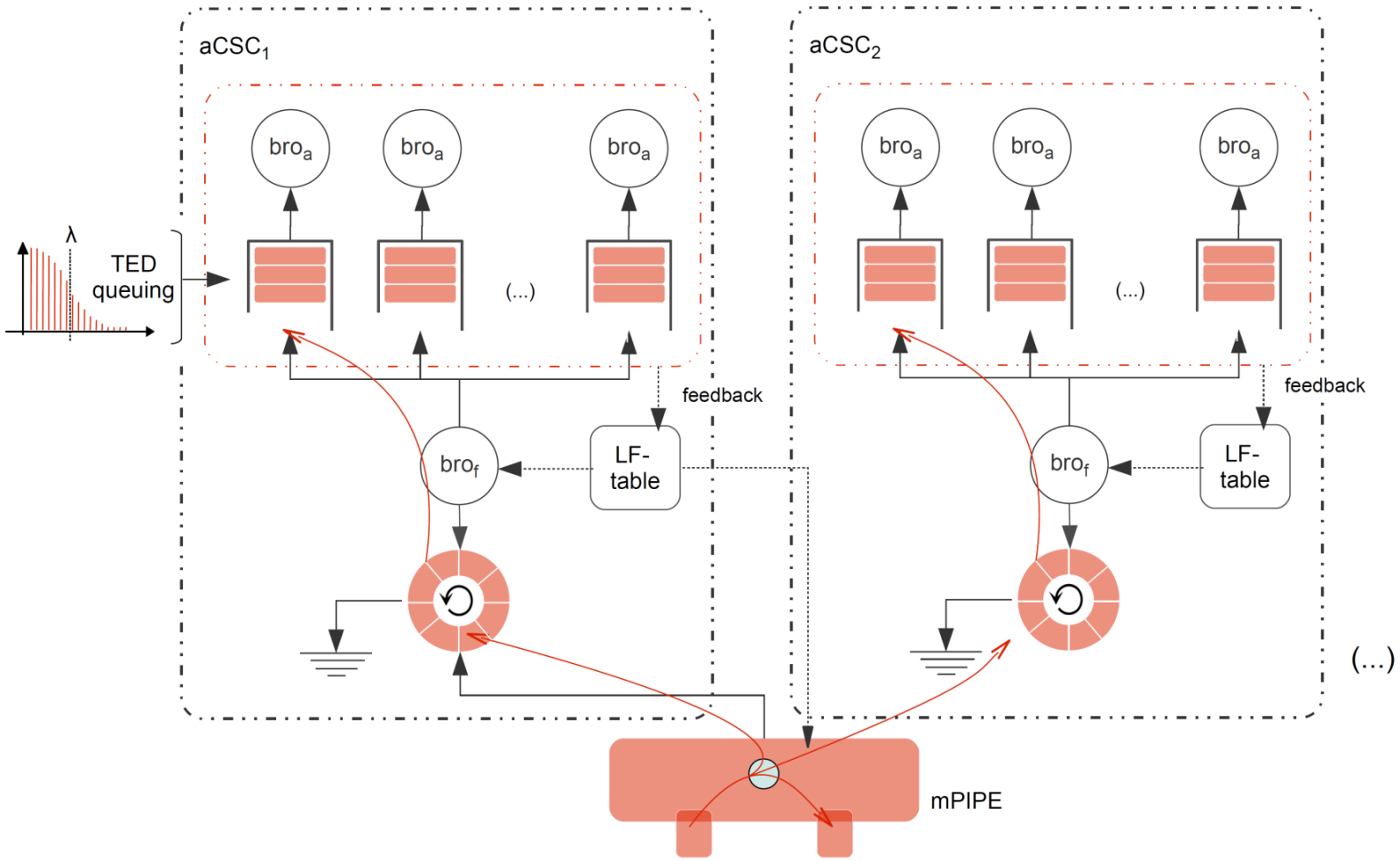
Packet Forwarding and Packet Analysis: Two Faces of the Same Coin (Ex Ante and Ex Post Configurations)

- Optimizing for {Speed + Analysis} != Optimizing for Speed + Optimizing for Analysis
- Optimality is achieved with the joint problem

- aCSC cells -- a network configuration to address the jointly optimized solution:



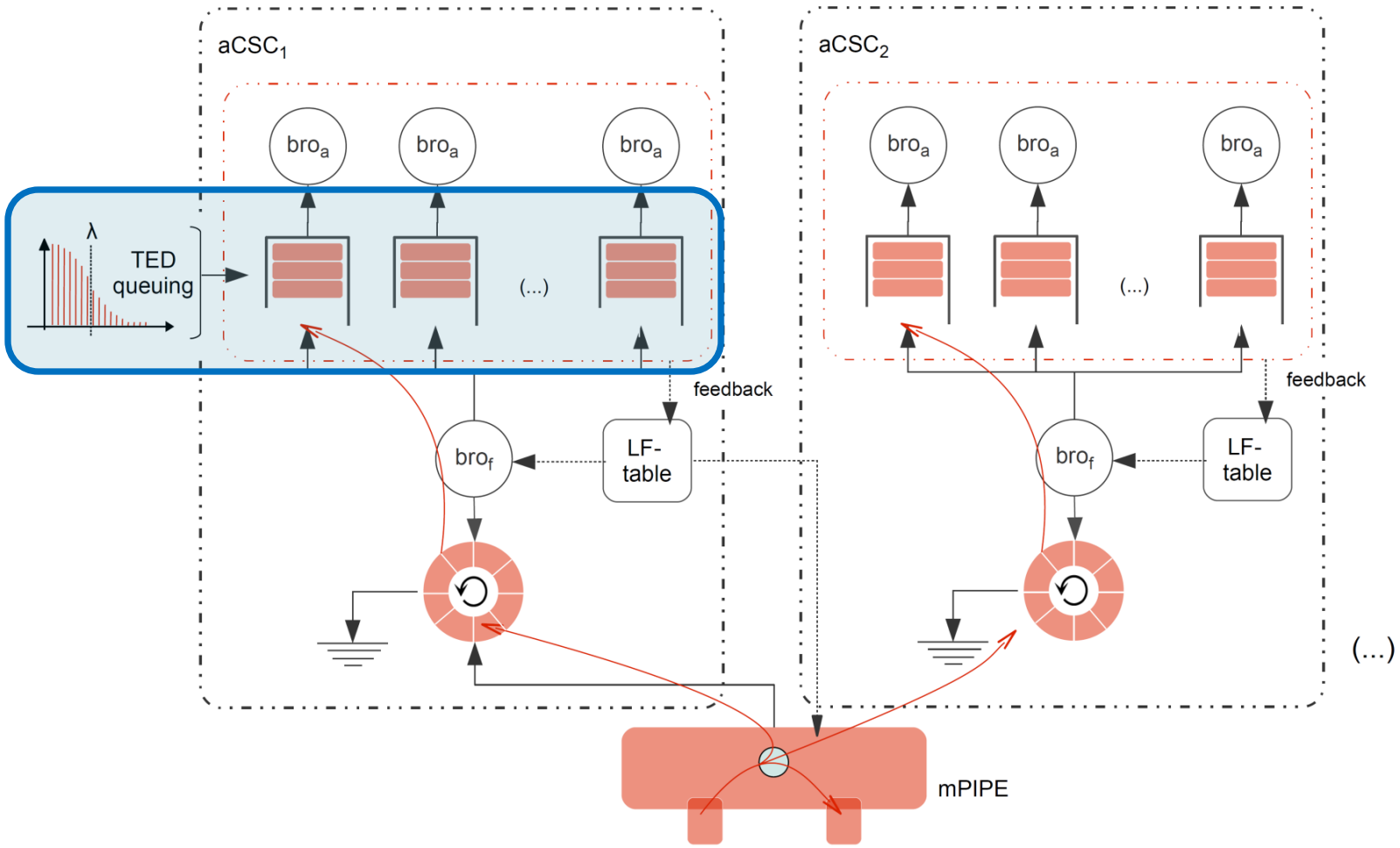
Mcore: Mapping Architecture onto a Manycore



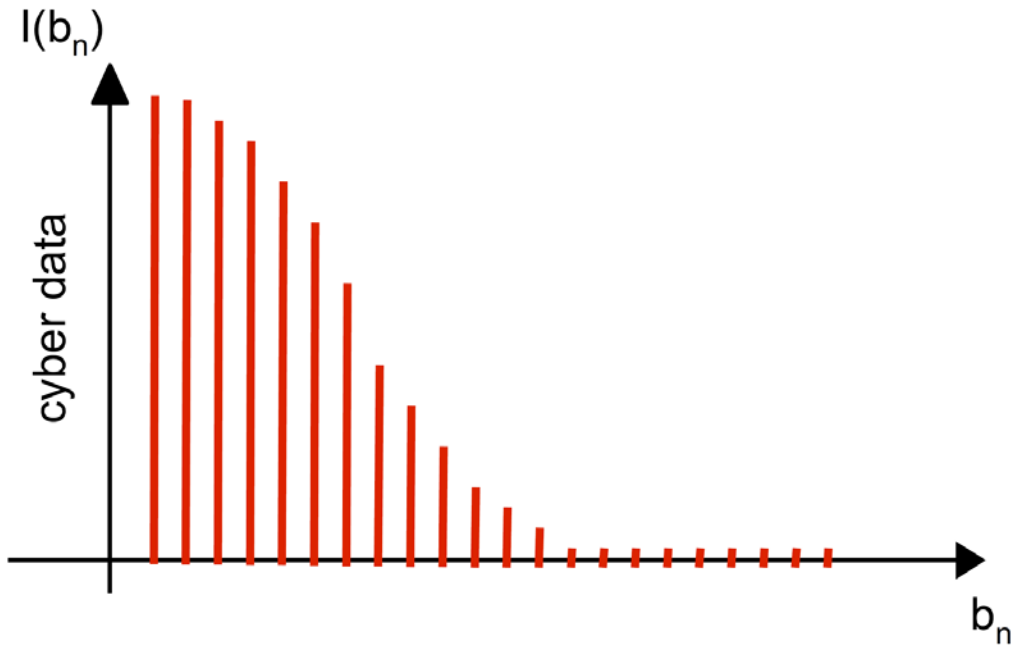
High-Performance Features

Feature:	Utility:
mPIPE: Programmable packet classification and switching engine with up to 80G bps and 120M packets-per-second of throughput	- High-speed packet switching - No kernel involvement
TED Queuing (Tail Early Dropping): Intelligent congestion-avoidance packet dropping policy.	- Intelligent packet dropping - Designed to make the system operate out of cache
LF - **Data Structure**: Lock-free data structure with low probability of false negative to convey feedback from A to F	- Zero locks everywhere - Minimize memory contention

TED Queuing


aCSC1
aCSC2
broa
broa
broa
broa
broa
broa
λ
TED
queuing
(...)
(...)
feedback
feedback
brof
brof
LF-
table
LF-
table
(...)
mPIPE

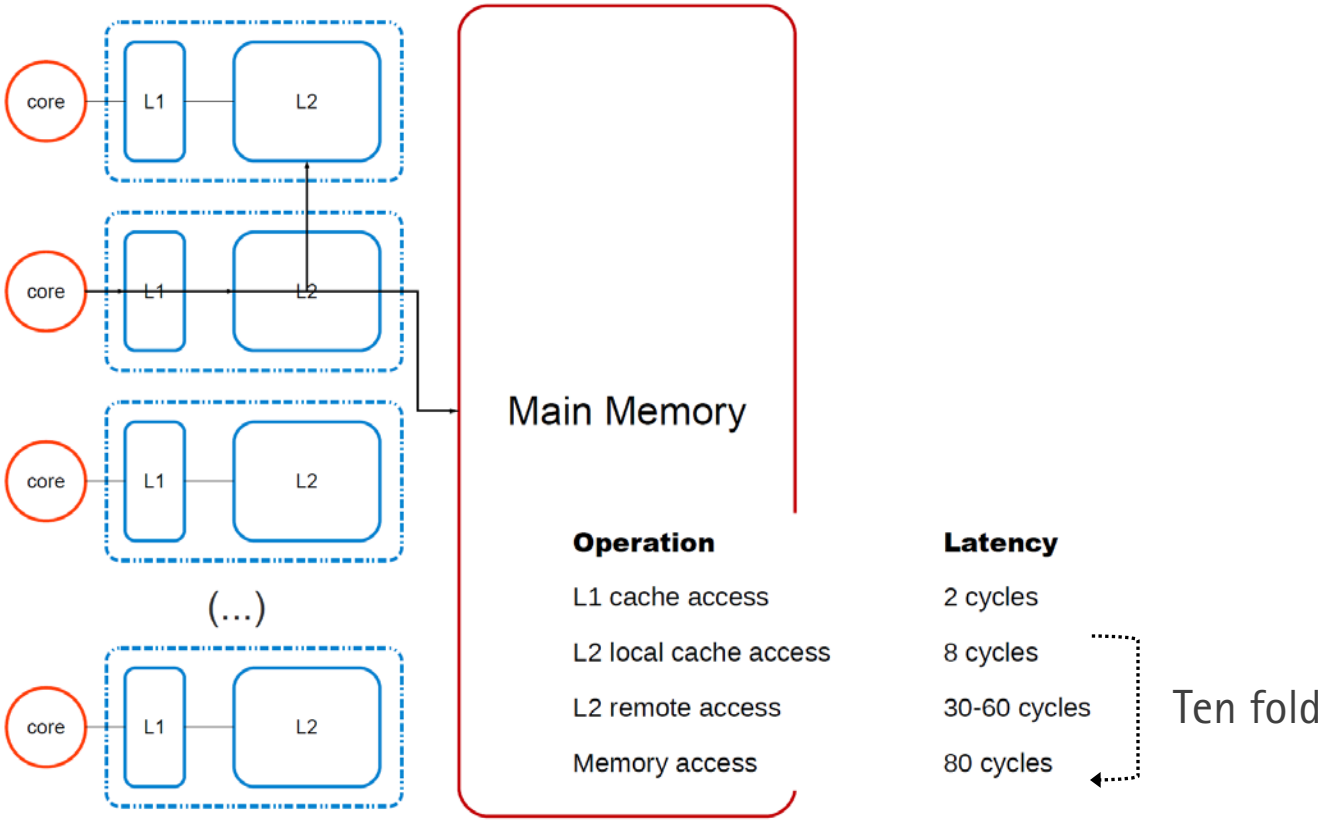
- Suppose that the system is so stressed that we need to drop a packet. In the context of CSC, which packet should we drop?
- Approach: leverage the flows' heavy tails



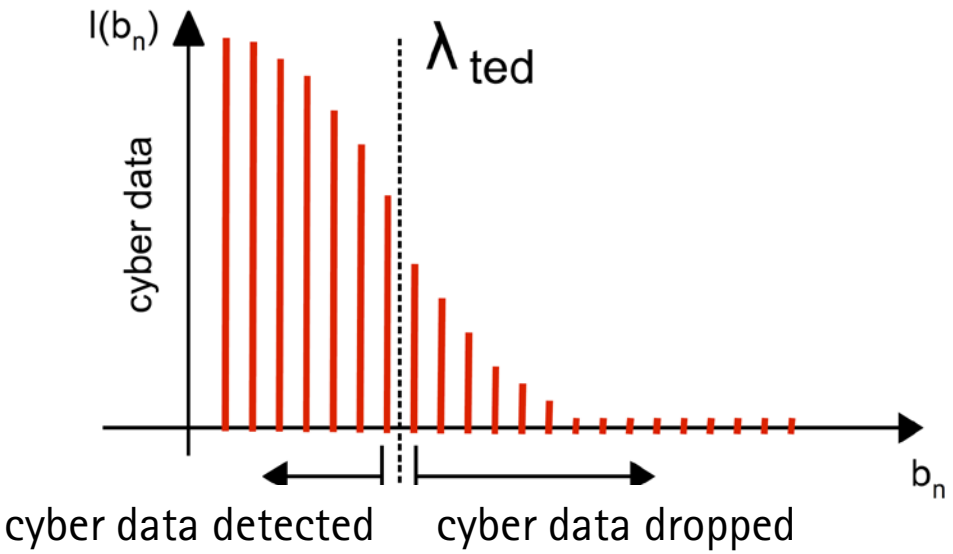
b_n: n-th bit received from a flow
$I(b_n)$: degree of cyber security information carried by bit n-th

TED Queuing

- TED Queuing designed to (1) exploit heavy tails and (2) to find an optimal trade-off in hierarchical memory architectures



- TED queuing principle:



Lemma 1: Let c_{in} be the amount of cyber data per unit of time received by the node and let π_c be the amount of cyber data per unit of time that it can actually process:

- If $\pi_c < c_{in}$, there exists a value λ_{ted}^{h} such that for any $\lambda_{ted} > \lambda_{ted}^{h}$, the analyzers in the node will be accessing packets from memory (cache miss).
- For any value of π_c, there exists a value λ_{ted}^{l} such that for any $\lambda_{ted} < \lambda_{ted}^{l}$, the analyzers in the node will be accessing packets from cache (cache hit).

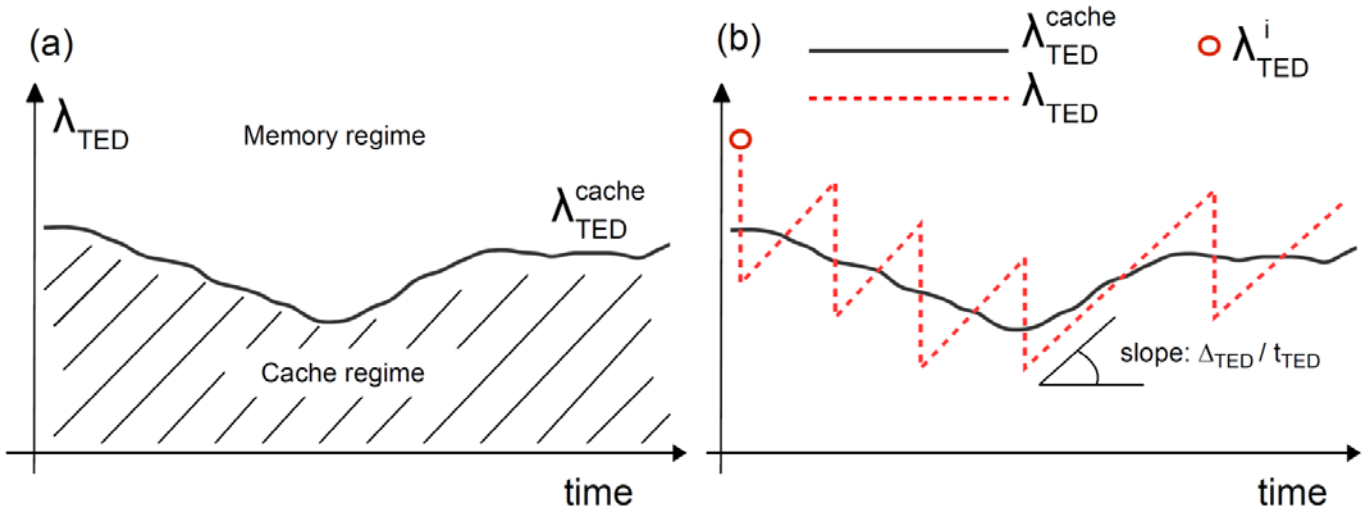
- TED queuing algorithm:

Constants: $\Delta_{ted}, t_{ted}, \lambda^{i}_{ted}$.

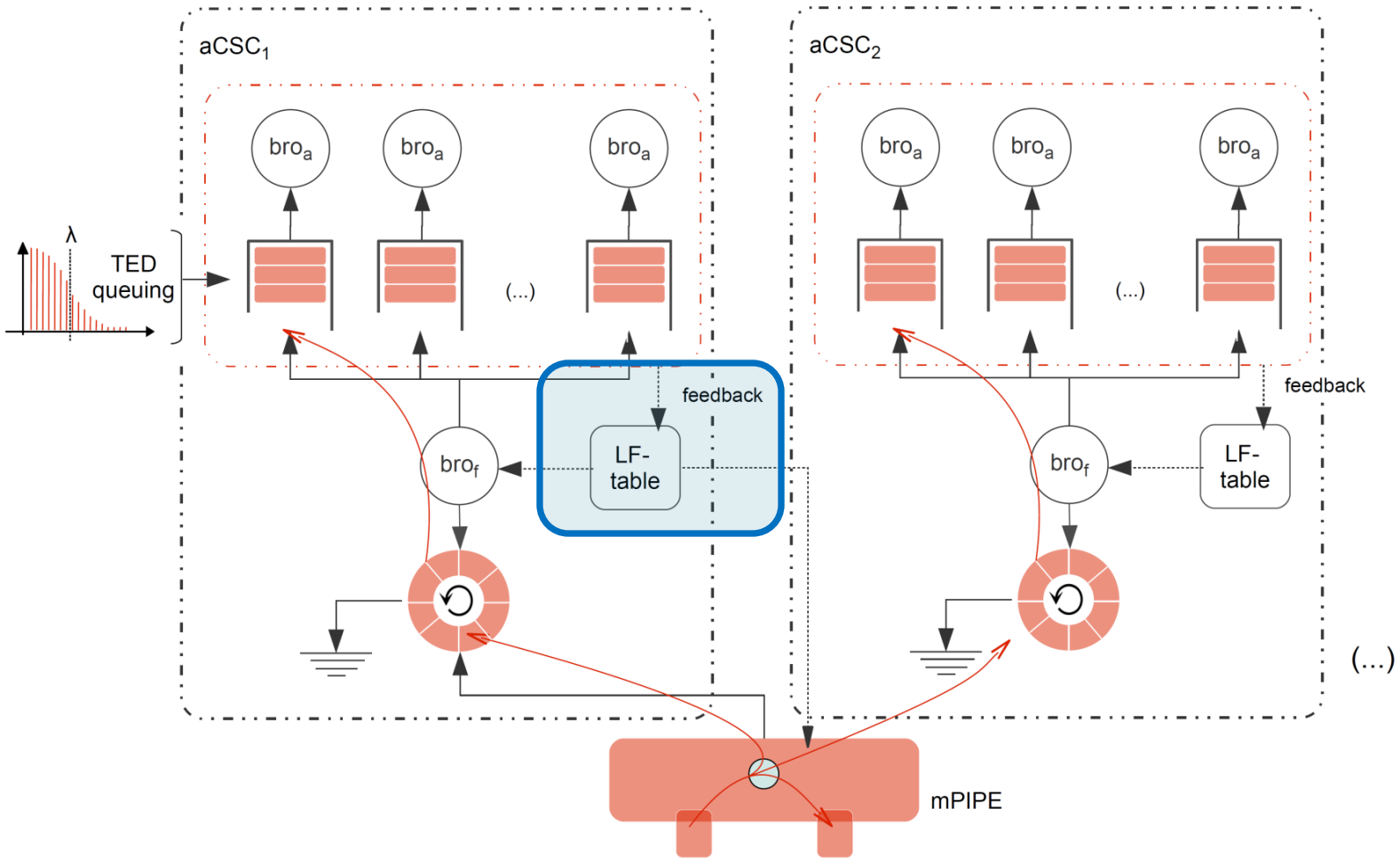
Step 1. Start with $\lambda_{ted} = \lambda^{i}_{ted}$, for an arbitrary λ^{i}_{ted}

Step 2. If the cell is operating in memory regime, then keep reducing λ_{ted} by a value $\Delta_{ted} > 0$ until the cell starts to operate in cache regime.

Step 3. Wait a period of time t_{ted} and then increase λ_{ted} by Δ_{ted}. After that, return to step 2.



LF - Data Structure



LF - Data Structure

- Data structure designed to satisfy the following specifications:
  - It must be able to keep at least one flag for each element stored.
  - It can be concurrently written and read by multiple writers and multiple readers but it does not require locks to preserve the correctness of its elements
  - It can tolerate a low probability of false negatives but not false positives

- Algorithm: Initial state: $\mathbf{T[e] = NULL}$ for all e such that $0 \leq e < N$;

  Writer (analyzer) algorithm:

  Upon detecting that c needs to be dropped, do:

  $T[h(id(c)) \ modulo \ N] = h(id(c))$;

  Reader (forwarder) algorithm:

  Upon receiving a packet from connection c, do:

  If $T[h(id(c)) \ modulo \ N] == h(id(c))$

  Drop the packet;

  Otherwise

  Forward the packet;

LF⁻ Data Structure

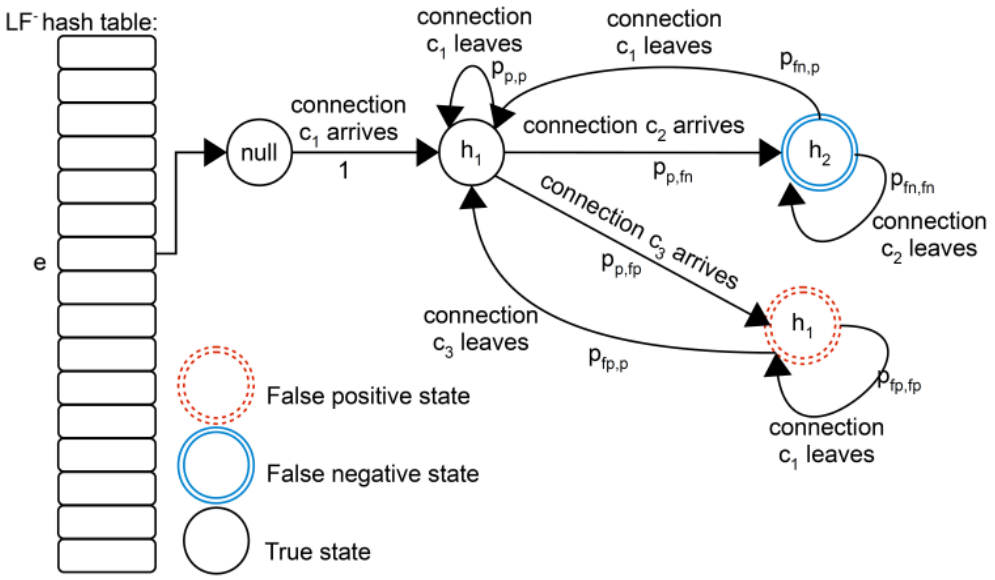
Lemma 1: LF^- state correctness. An element in a data structure constructed using the LF^- algorithm is in a positive state with probability p_t, in a false negative state with probability p_{fn}, and in a false positive state with probability p_{fp}, where $p_{fp} \ll p_{fn} \ll p_t$ and $p_t \approx 1$.

Proof. [Working out the math—see HPEC12 paper]

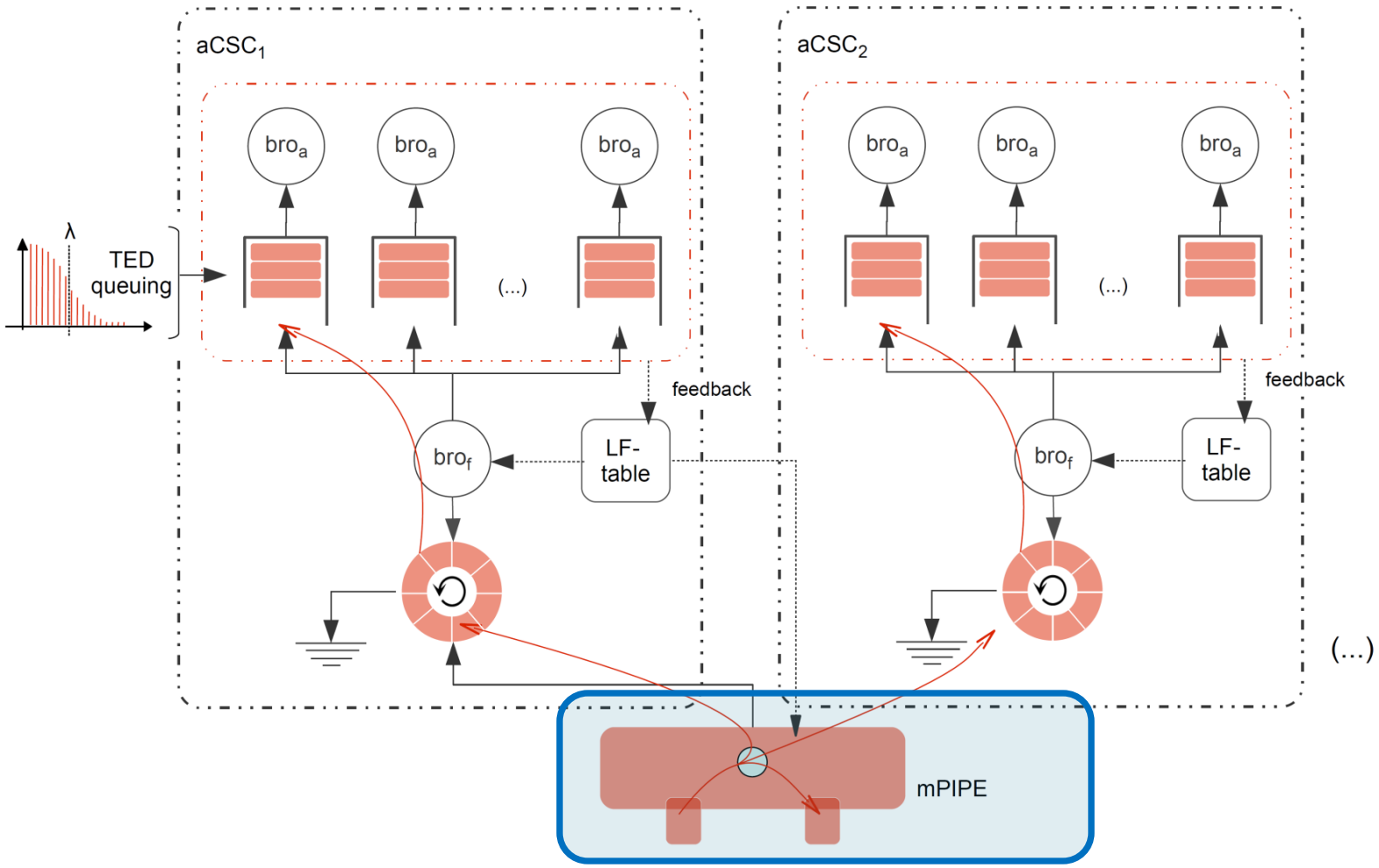
$$p_t \approx \frac{1}{\left(1 + \frac{1 - p_{t,t}}{p_{fn,t}}\right)}$$

$$p_{fn} \approx \frac{p_{t,fn}}{(1 - p_{fn,fn})}$$

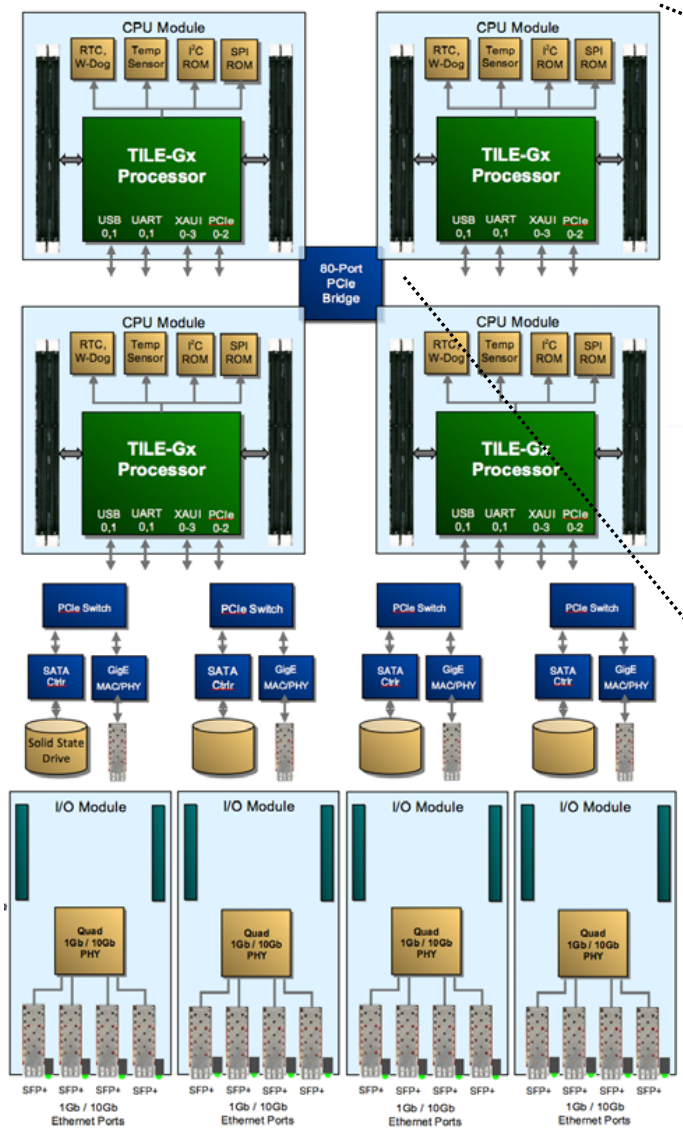
$$p_{fp} \approx \frac{p_{t,fp}}{(1 - p_{fp,fp})}$$



Tilera Gx

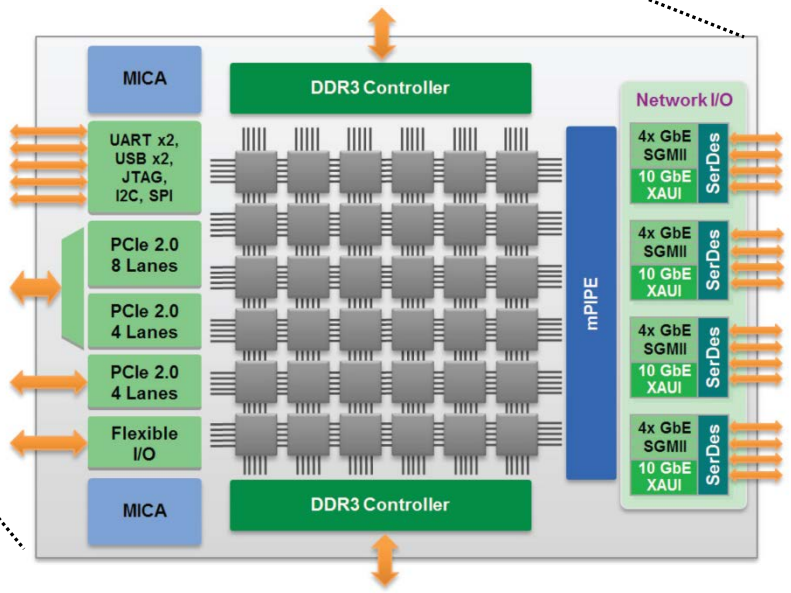

aCSC1
aCSC2
broa
broa
broa
(...)
λ
TED
queuing
feedback
brof
LF-
table
(...)
mPIPE

Tilera Gx (TILExtreme)

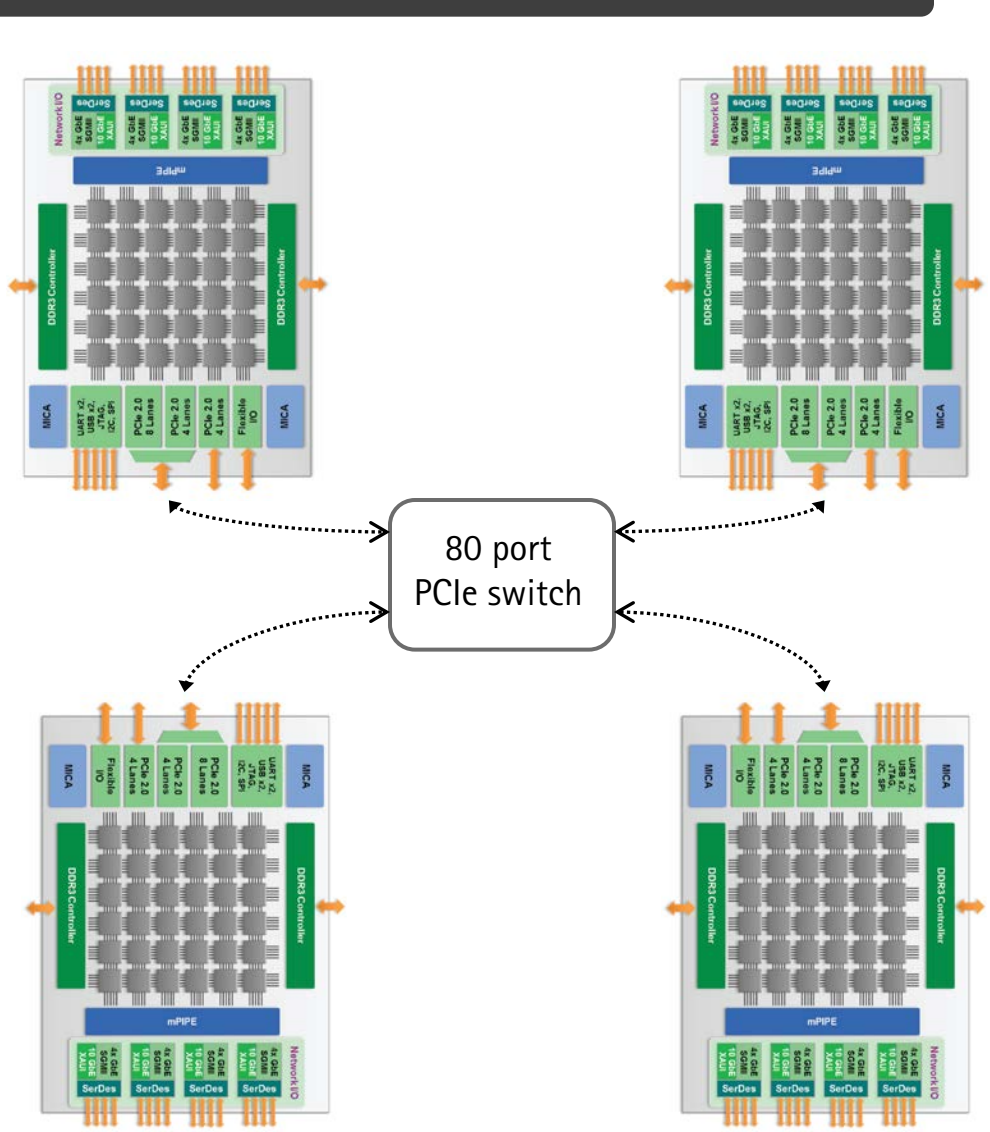


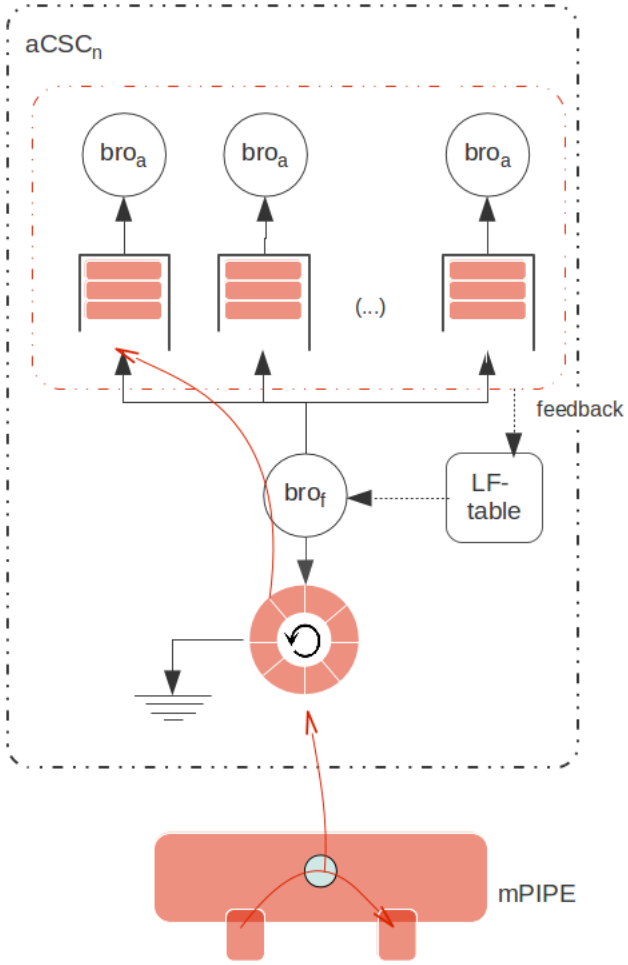
Manycore platform:

- 36 tiles per processor x 4
- 4 x 10Gbe per processor
- Roadmap for 100 tiles per processor
- mPIPE: I/O acceleration
- MiCA: crypto acceleration

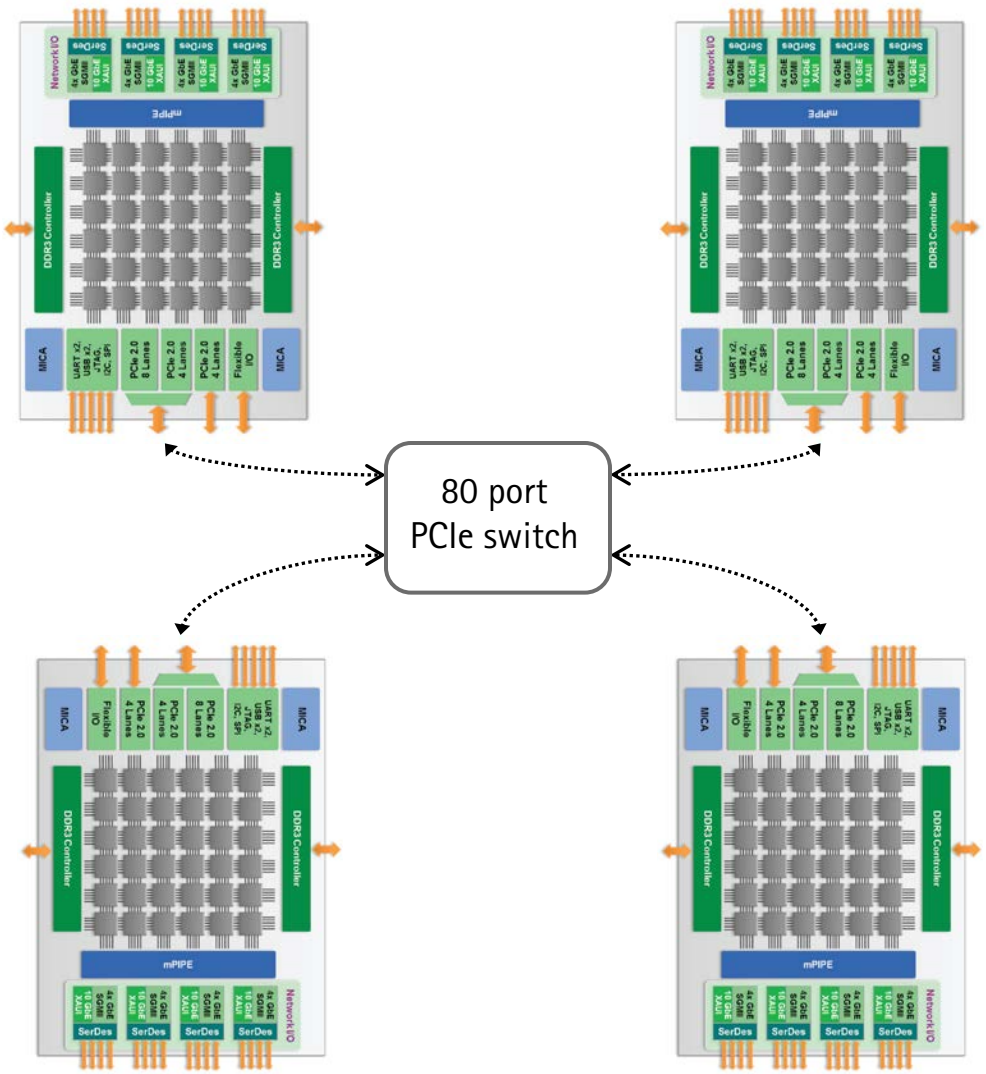


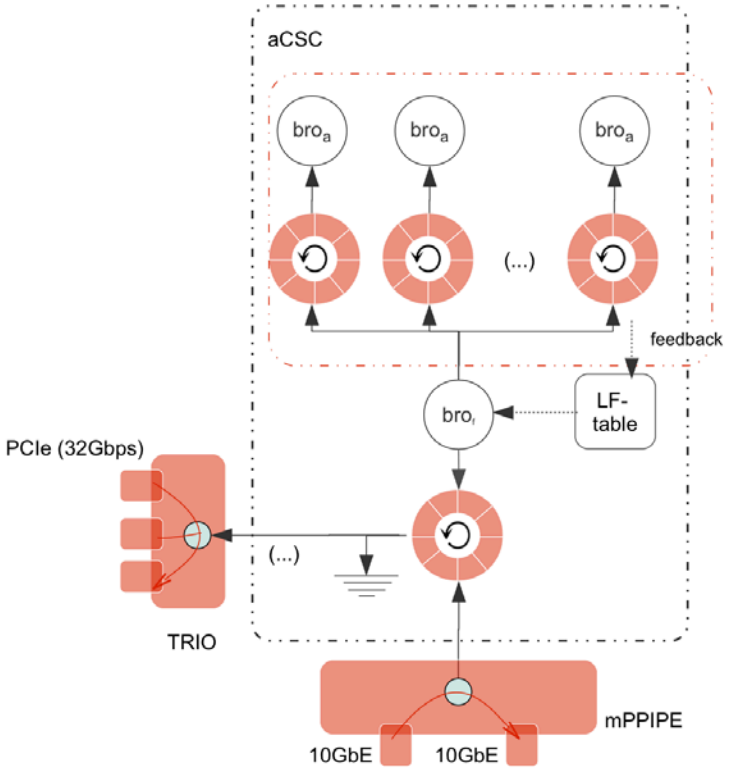
Tilera Inter-Processor Communication



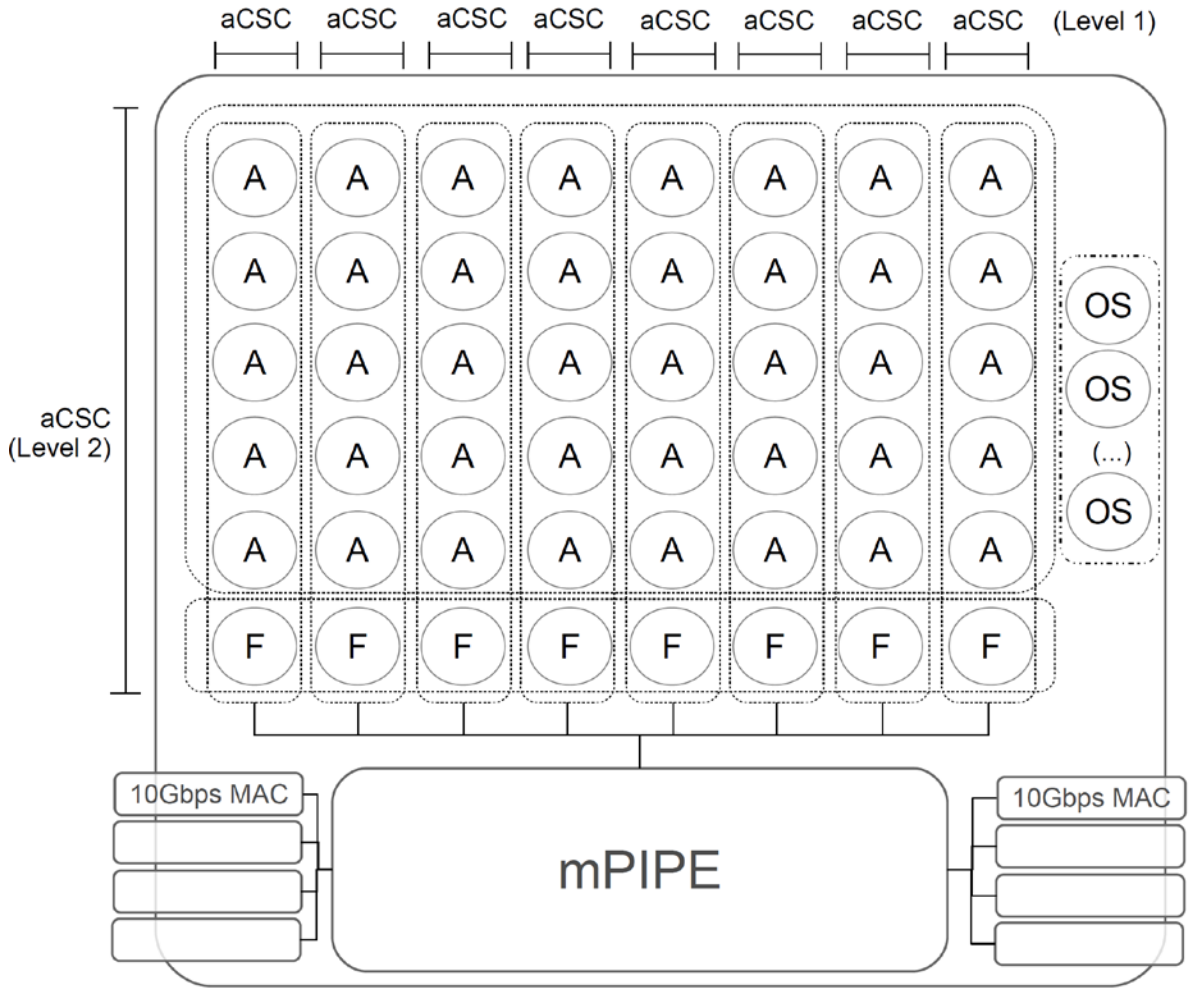


Tilera Inter-Processor Communication





Mcore mapping



Mcore extensions for Bro

mcore-def.bro

```
type mcore_node_type: enum { NULL, FORWARDER, ANALYZER, FORWARDER_ANALYZER };
type mcore_node_id: count;

const mcore_interfaces_input: set[string] = {
} &redef;

const mcore_interfaces_output: set[string] = {
} &redef;

const mcore_do_shunting = F &redef;
const mcore_ted_flow_threshold = 2000 &redef;  # in number of bytes within a flow
const mcore_ted_queue_threshold = 50  &redef;  # in % of queue utilization from 0 to 99
const mcore_force = F &redef;

const mcore_enabled = F &redef;
const mcore_is_manager = F &redef;

const mcore_node_map: table[mcore_node_id] of mcore_node_type = {
} &redef;
```

Mcore extensions for Bro

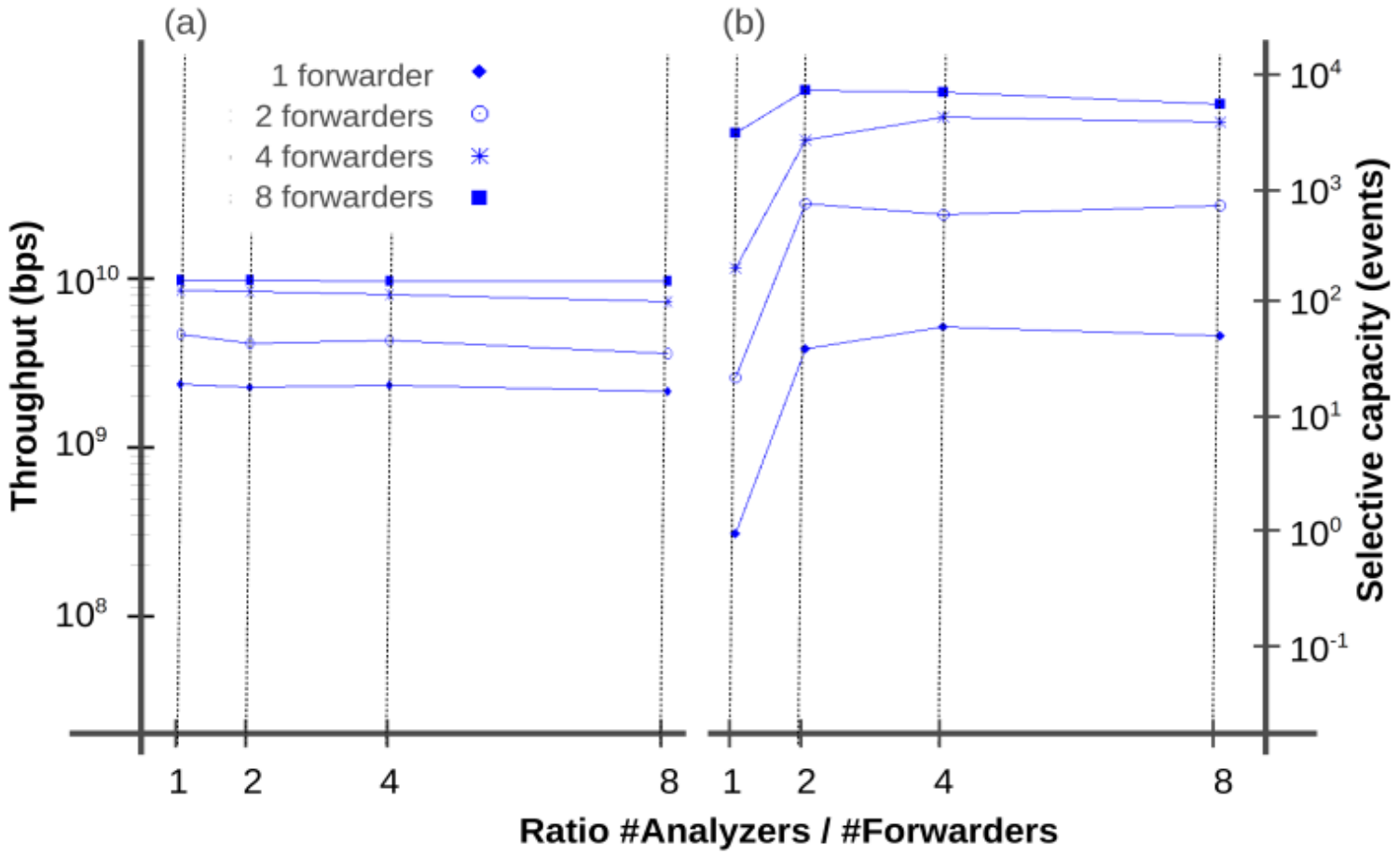
mcore-x-y-z.bro

```
redef mcore_node_map = {
        [0]           = FORWARDER,
        [1]           = FORWARDER,
        (...)
        [x-1]         = FORWARDER,
        [x]           = ANALYZER,
        [x+1]         = ANALYZER,
        (...)
        [x+y-1]       = ANALYZER,
        [x+y]         = FORWARDER_ANALYZER,
        [x+y+1]       = FORWARDER_ANALYZER,
        (...)
        [x+y+z-1] = FORWARDER_ANALYZER,}
;
```

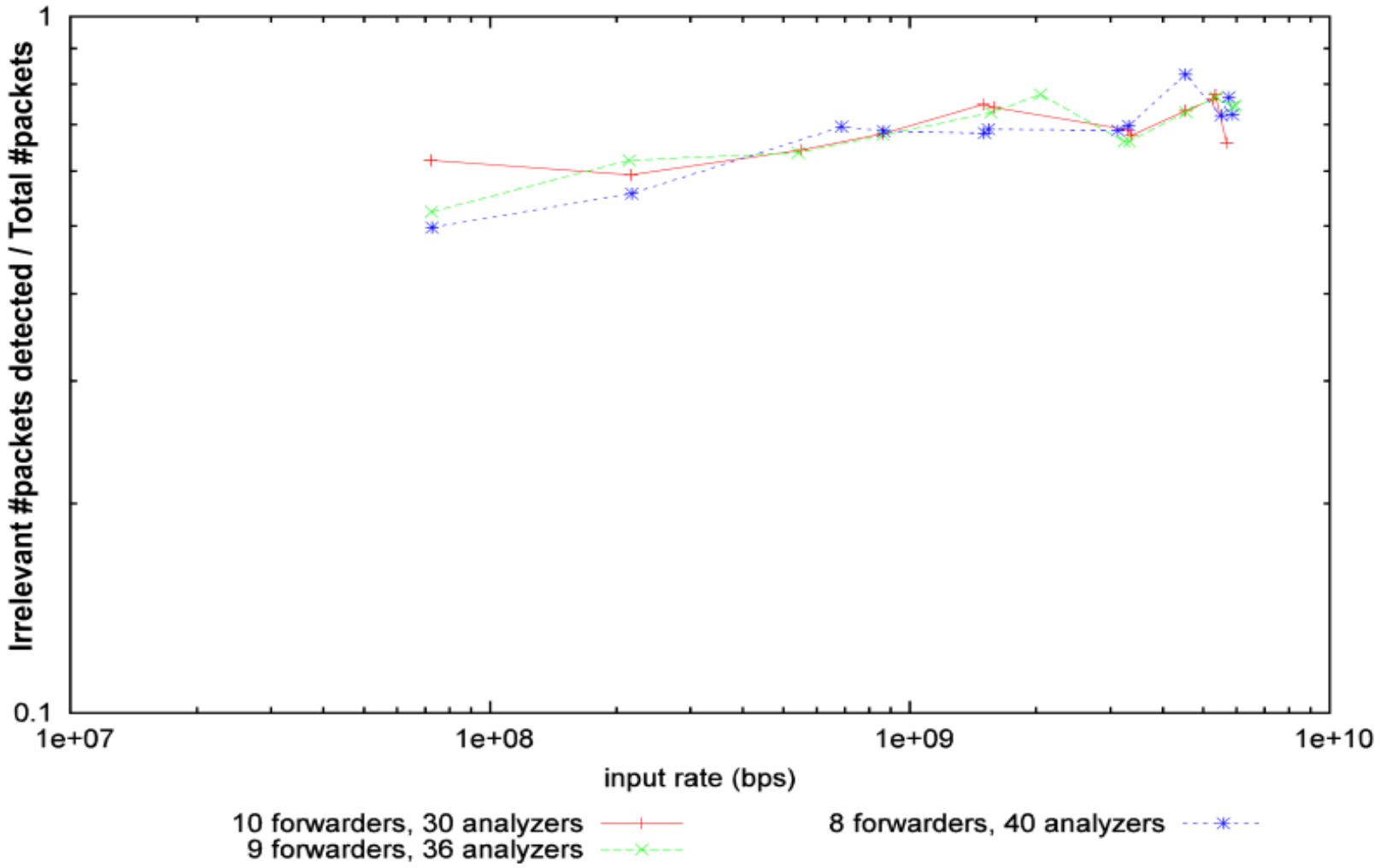
arch/tilera/mcore.h

```
void mcore_get_ring_writer(void);
void mcore_get_ring_reader(void);
int mcore_conf_rings(int num_analyzers, int num_forwarders);
int mcore_init_rings(int NumAnalyzers);
int mcore_link_init(int *cpu_ranks,
                    int num_forwarders,
                    int num_rings,
                    char* if_input,
                    unsigned char mac_output[][ADDR_LEN],
                    char if_output[][MAX_STRLEN],
                    int num_output_if,
                    int num_input_if);
void * mcore_alloc_shmem(unsigned int size);
void mcore_free_shmem(void * mem, unsigned int size);
```

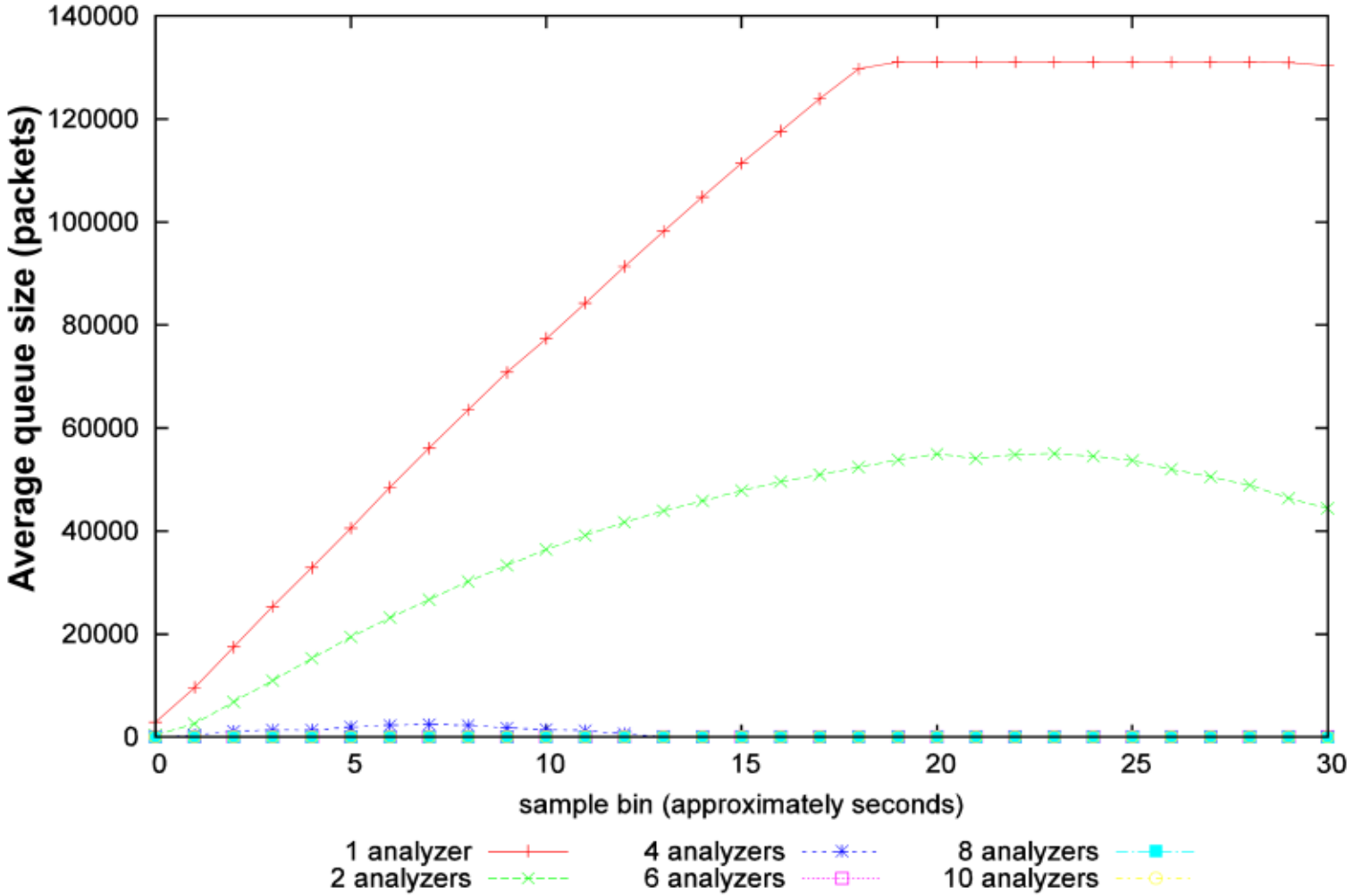
Performance (Tilepro)



Performance (Tilepro)



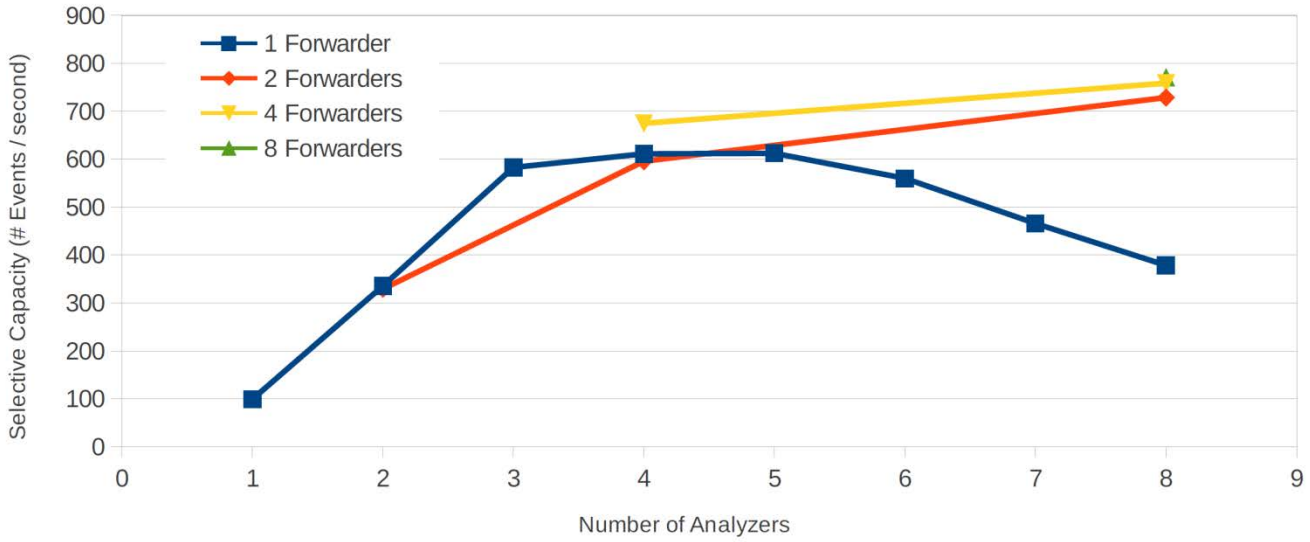
Performance (Tilepro)



Performance (Tile-GX)

Analytics: Bro's default set + http_track.bro

	# Forwarders	# Analyzers	Selective Capacity (events/sec) Per Tile	Selective Capacity (events/sec) Total	Forwarding throughput (Gbps)
1 Forwarder	1	1	98.920041	98.920041	9.8
	1	2	167.699023	335.398046	9.8
	1	3	194.169252	582.507756	9.8
	1	4	152.70955	610.8382	9.8
	1	5	122.405928	612.02964	9.8
	1	6	93.275352	559.652112	9.7
	1	7	66.541857	465.792999	9.7
	1	8	47.263248	378.105984	9.7
2 Forwarders	2	2	164.556612	329.113224	9.8
	2	4	148.848772	595.395088	9.8
	2	8	91.044497	728.355976	9.8
4 Forwarders	4	4	168.595005	674.38002	9.8
	4	8	94.797063	758.376504	9.8
8 Forwarders	8	8	96.312077	770.496616	9.8



Going Forward

- Live demo at the SCinet Research Sandbox this coming November → target: 4 x Tile Gx-36 from 10 to 80 Gbps.
- Tilera Gx roadmap toward 100 core processors. Leverage this roadmap and other development in the manycore space toward scaling to terabit computing.
- Looking for testbeds that may be interested in testing Mcore technology.

References

- S. Campbell, J. Lee, "Intrusion Detection at 100G", The International Conference for High Performance Computing, Networking, Storage, and Analysis, November 14, 2011
- "Department of Energy Builds National 100GigE Research Net," Press Release, July 13, 2011.
- M. Vallentin, R. Sommer, J. Lee, C. Leres, V. Paxson, B. Tierney, "The NIDS Cluster: Scalable, Stateful Network Intrusion Detection on Commodity Hardware," Recent Advances in Intrusion Detection, 2007
- National Institute of Standards and Technologies, "The NIST Definition of Cloud Computing," Special Publication 800-145, September 2011.
- J. Ros-Giralt, P. Szilagyi, R. Lethin, "Scalable Cyber-Security for Terabit Cloud Computing (Extended Version)," Reservoir Labs Technical Report, May 2012.
- V. Paxson, "Empirically derived analytic models of wide-area TCP connections," IEEE/ACM Transactions on Networking, 2(4):316–336, August 1994.
- Jose Gonzalez, Vern Paxson, Nicholas Weaver, "Shunting: A Hardware /Software Architecture for Flexible, High Performance Network Intrusion Prevention," ACM Conference on Computer and Communications Security, November 2007.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, "Introduction to Algorithms," The MIT Press; 3rd edition, 2009.
- J. Ros-Giralt, P. Szilagyi, "A Lockless Hash Table With Low False Negatives," mathematical note, Reservoir Labs, April 2012.
- Floyd, Sally; Jacobson, Van. "Random Early Detection (RED) gateways for Congestion Avoidance". IEEE/ACM Transactions on Networking, 1(4): 397–413, August 1993.
- Vern Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," Computer Networks, December 1999.
- M. Berezecki, E. Frachtenberg, M. Paleczny, K. Steele, "Many-core key-value store," igcc, pp.1-8, 2011 International Green Computing Conference and Workshops, 2011.
- "Tilera Unveils the Ultimate Cloud Computing Processor," Tilera Press Release, June 2011.
- Carl Ramey, "TILE-Gx100 ManyCore Processor: Acceleration Interfaces and Architecture," Tilera Corporation, August 2011.

Testbed in NYC Labs