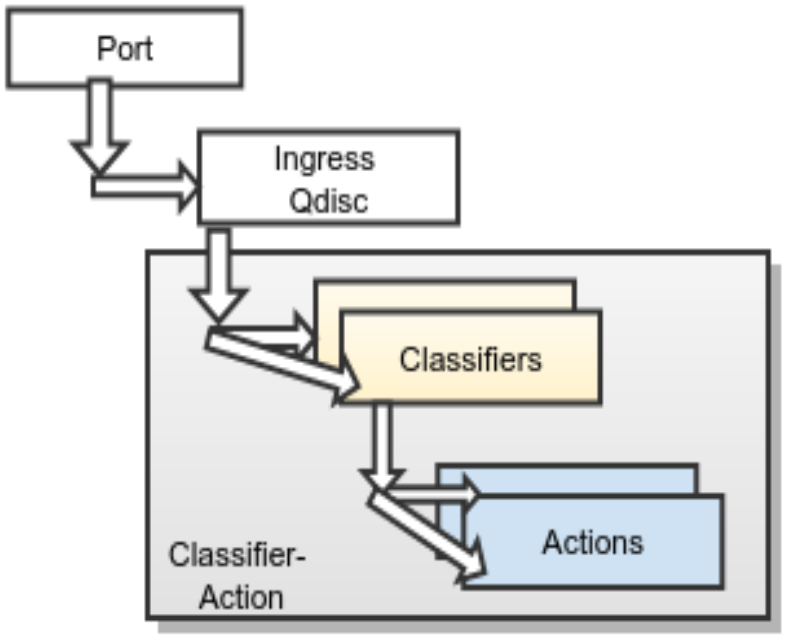
The CLASShoFIRES: Who's Got Your Back?

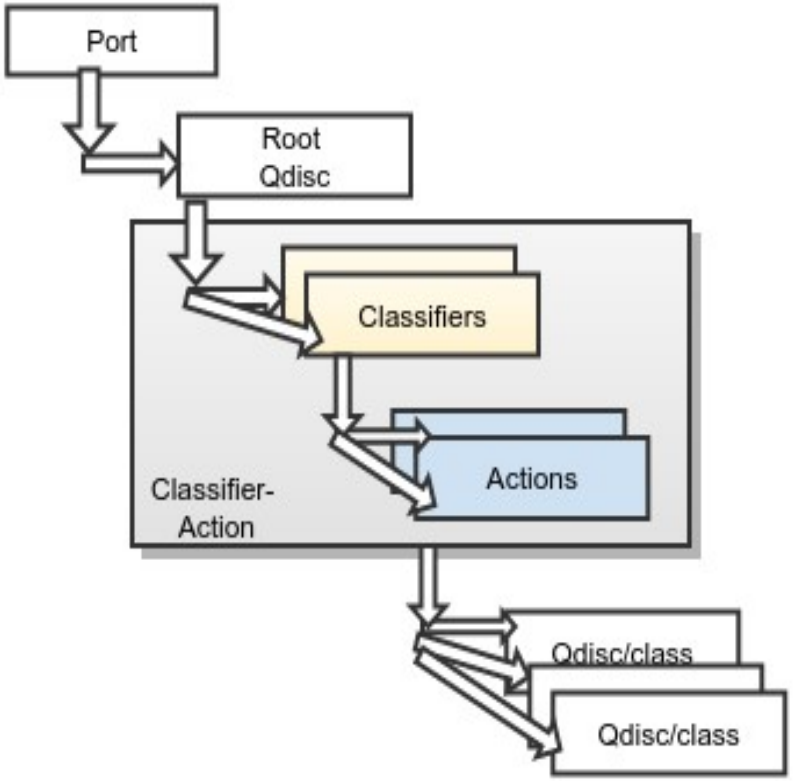
Jamal Hadi Salim
Lucas Bates

Linux Traffic Control: Overview

Ingress hook

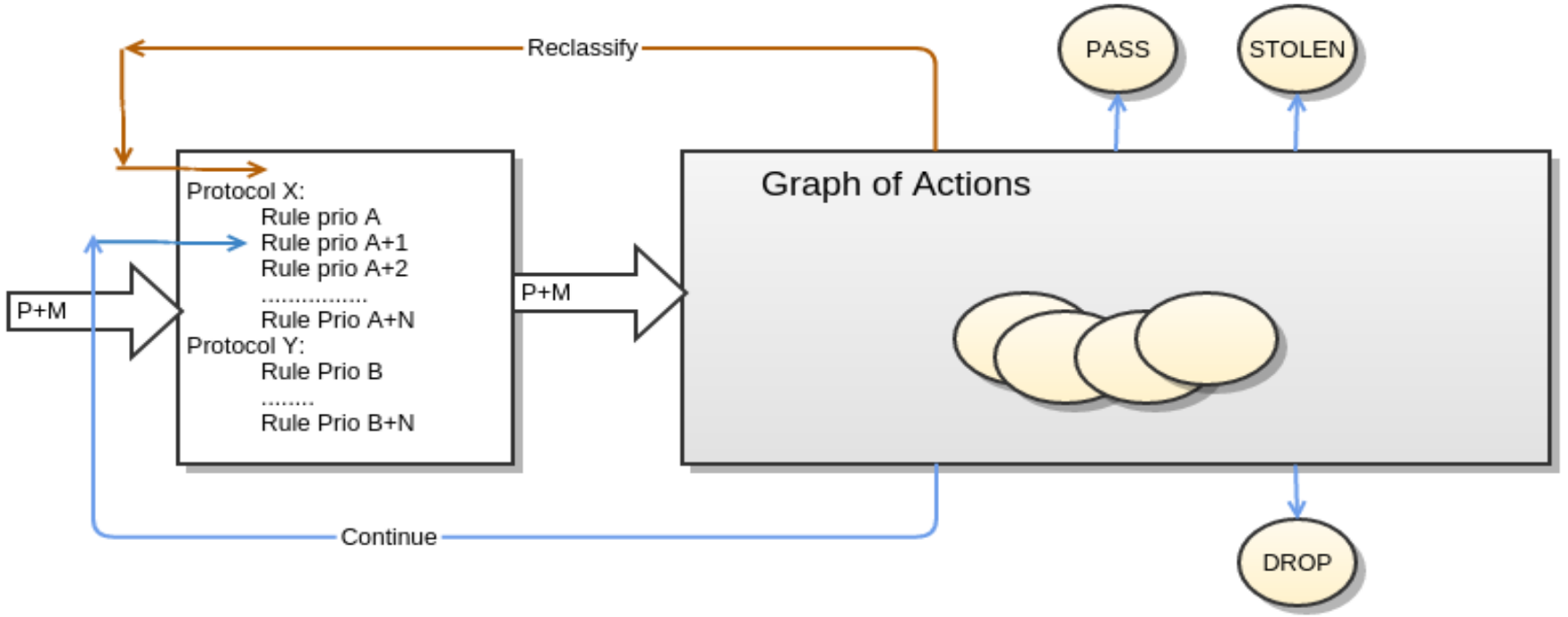


egress hook



new additional classact hook at the egress port

The Lord of All Rings: TC Classifier Action Subsystem



- Sorry Sauron, No *ONE ring* to rule all Classifiers
  - Unix philosophy: Do one thing and do it well
    - Can add new classifiers and actions
    - Formal BNF grammar for describing policy composition
      - Allows loops and branches

Motivation

- Two new Classiffiers over the last year
  - *Flower* and *[e]bpf*
- Curiosity about perf numbers for these
  - Last time was in 2005 at UKUUG ;->
- Look like an easy netdev11 paper
  - Speaking of being delusional!
- Years of SDN “flow” work with big ASICs
  - Curious how much could be done within the kernel
- The DPDK/user-space processing noise

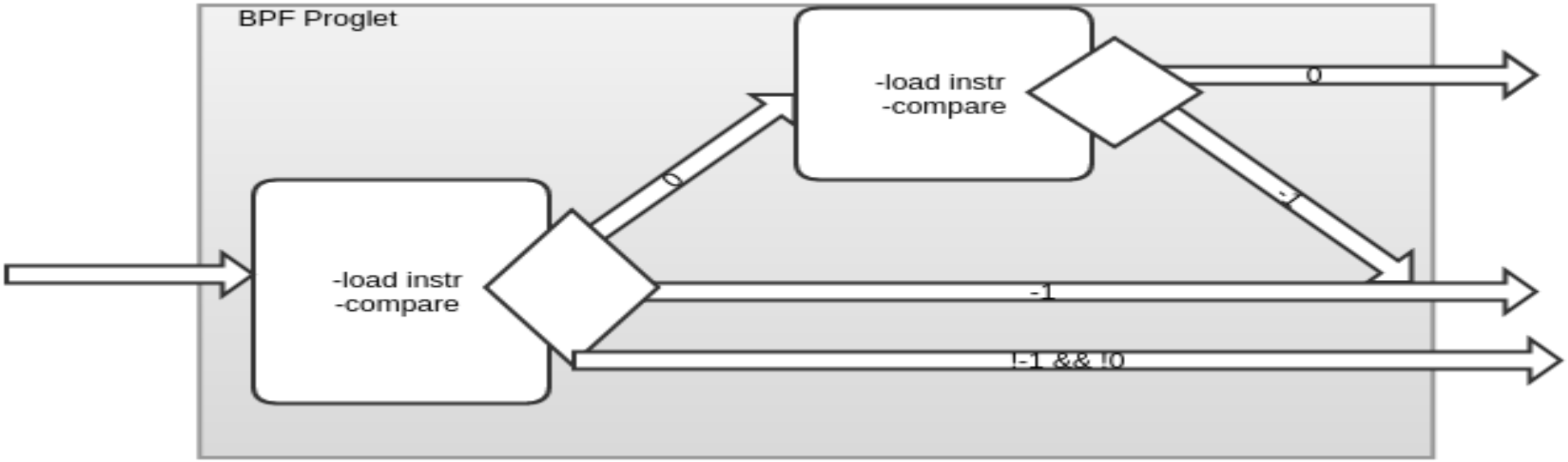
Picking the System Under Test

- Want to compare against what we do on ASICs:
  - Per-flow actions
    - At minimal accounting of bytes/packets
    - No hashing or groupings of flows via masks
      - That would be really boring
- Being Fair to all is important
  - Try to be objective and specify assumptions
  - Compare not Oranges with Apples

Picking the System Under Test

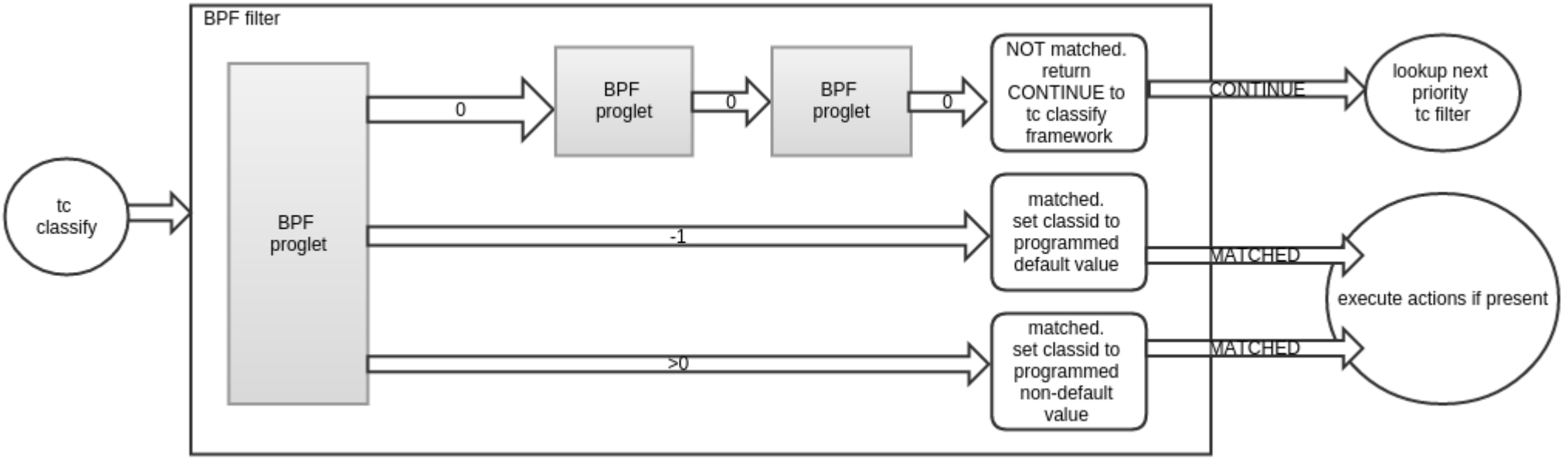
- In addition to *flower* and classic *bpf* pick an existing classifier for comparison
  - Picked the *u32* classifier
    - Swiss-army knife of packet filtering
      - Mysterious to some
        - So fun was to be had..

Classic bpf the Linux Way



- On Linux: bpf-based linux packet filter
  - Extended branching and not depending on netdev/port
- Note: This talk is not about using bpf as actions or extended bpf
- Restricted byte code compiled in user space intepretted in the kernel
  - Register based VM in the kernel means natural mapping to hardware cpu instructions
    - JIT BPF was fitting progression
  - Forward DAG

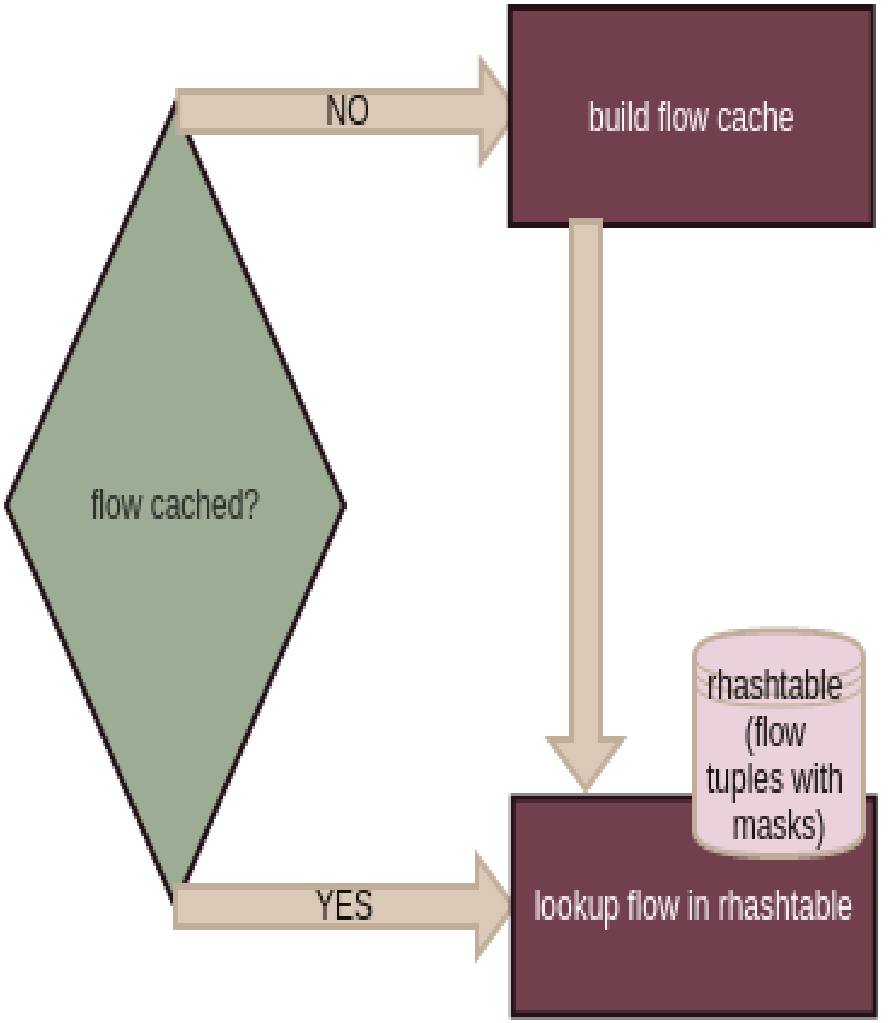
The bpf classic classifier



- Authored by Daniel Borkmann
  - Re-uses existing bpf-based linux packet filter
    - Extended branching and not depending on netdev/port
- Note: This talk is <u>not about using bpf as actions or extended bpf</u>
- Classical BPF had a big monolithic filter blob with limited program space
  - tc framework
    - allows many small bpf blobs
    - Ability to create policy loops with many small bpf blobs

The flower classifier

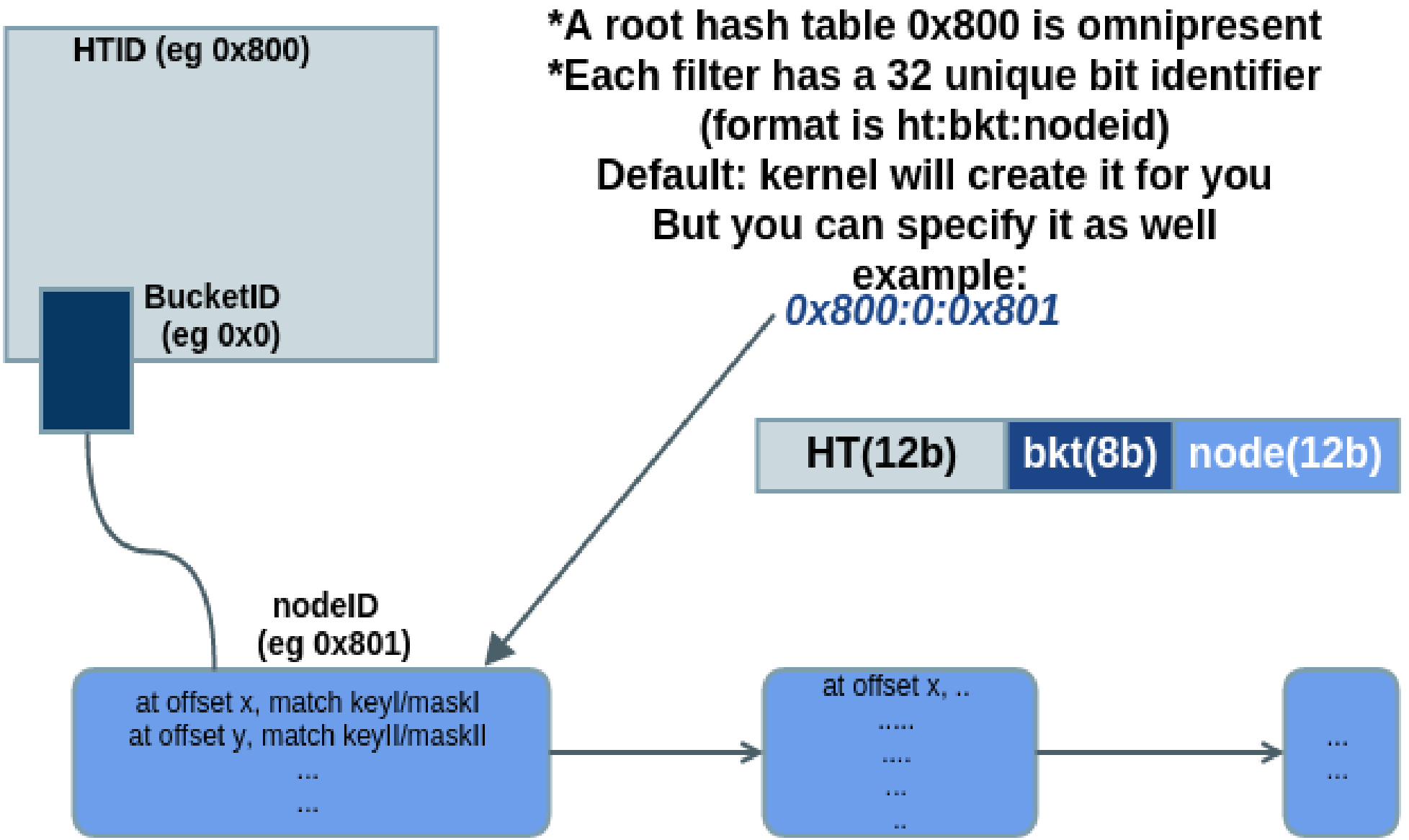
- Written by Jiri Pirko
- Clever: Utilizes commodity features
  - Flow cache
    - Already being built as packet traverses stack
  - Rhashtable
    - Optimized key-based hash table implementation



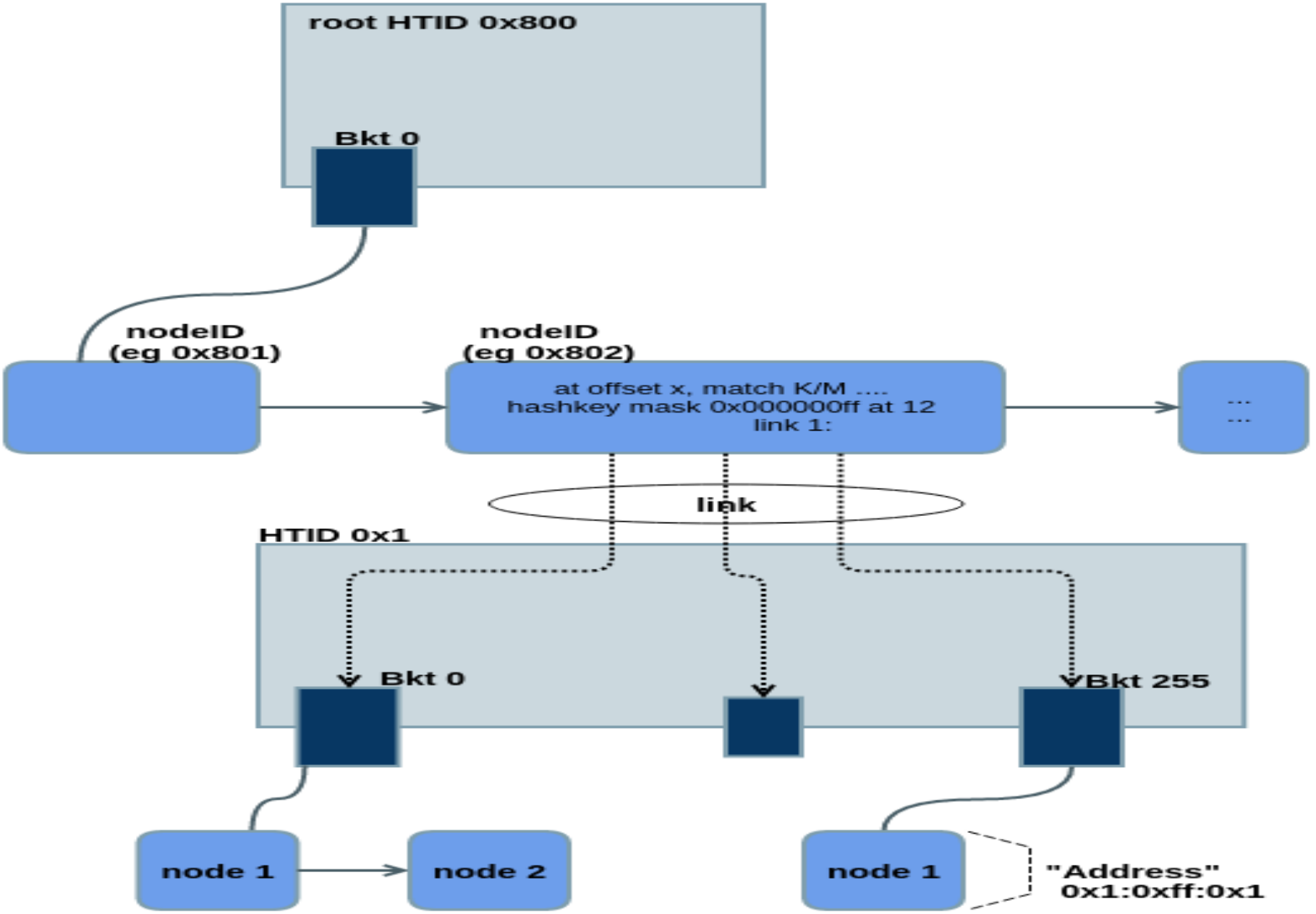
The flower classifier

- Currently supports a subset of flow tuples
  - Src/dst MAC, ethertype, src/dst Ipv4/6, src/dst transport port, ingress dev at egress
- Potential for rest of flow cache
  - Vlanid, MPLS labels, GRE keys, TIPC
  - New fields added over time
- Very human friendly
  - Both from cli and programmatic level

The u32 classifier



The u32 classifier



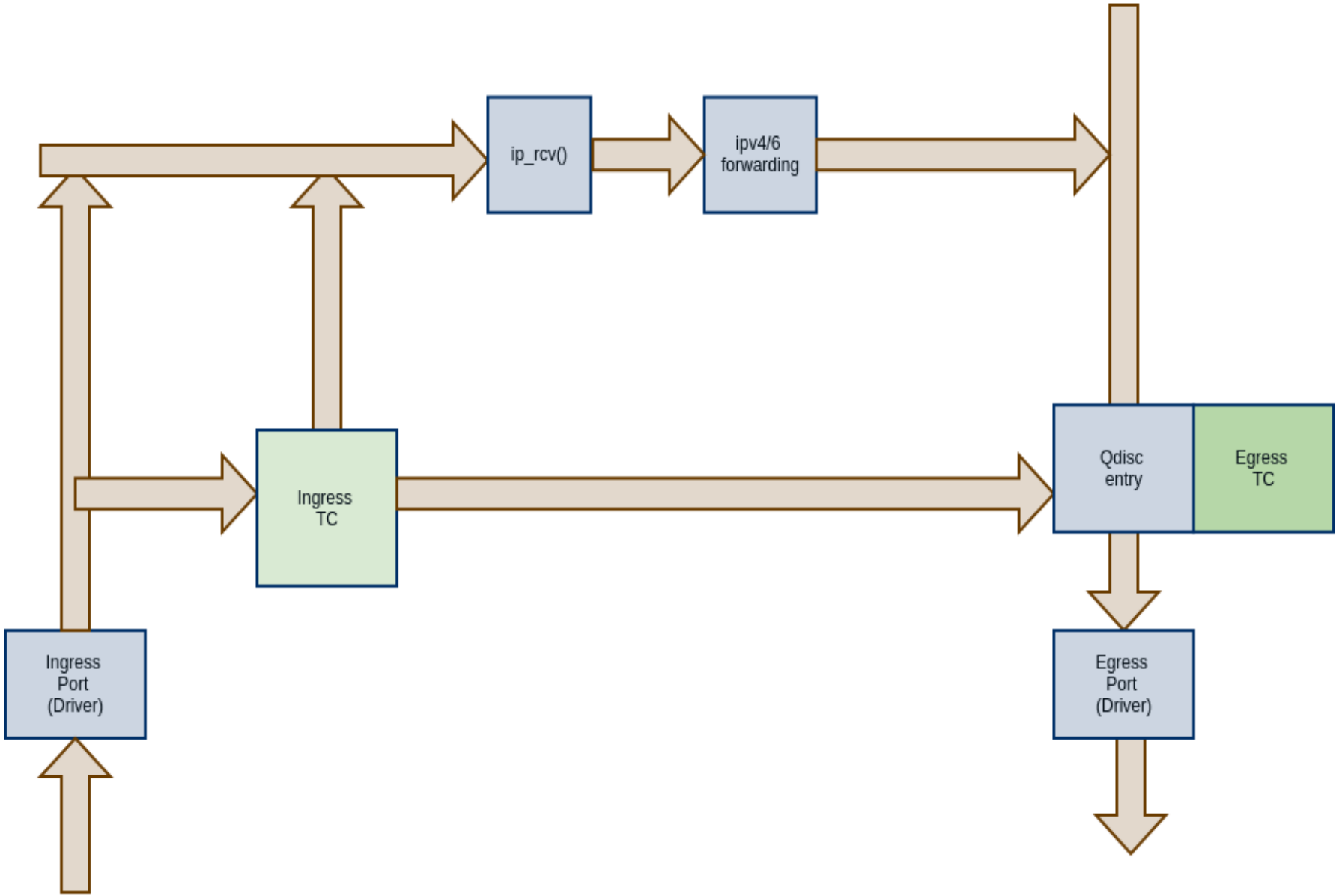
Picking The Metrics

- Datapath Throughput performance
  - Easy one we tackle here
- Datapath Latency
  - Unfortunately didnt have time to pursue this
- Usability
  - This is subjective if you care about humans
- Extensibility
  - Programmability
    - Either via scripting or coding
- Control path throughput and latency
  - Didnt have time to chase
    - Have opinions - will handwave in this talk

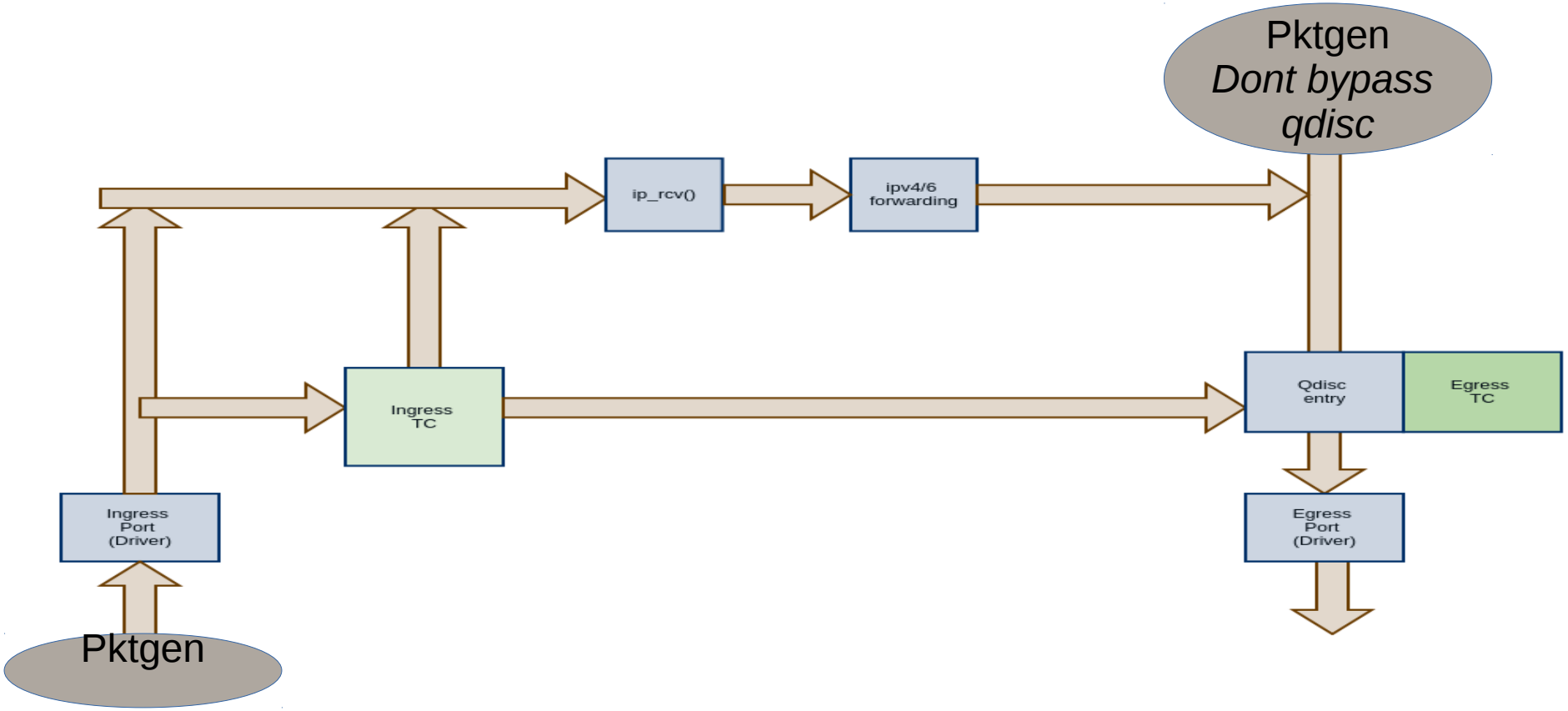
Focus On The SUT, kid

- Reduce as many variables as possible on System Under Test
  - SUT: implementation of classifier
- Possible distractions
  - System multi-processing locks
  - Driver code paths (both ingress + egress)
  - Slow system code paths
  - Intermediate handoff queues

Picking The Battle Scene

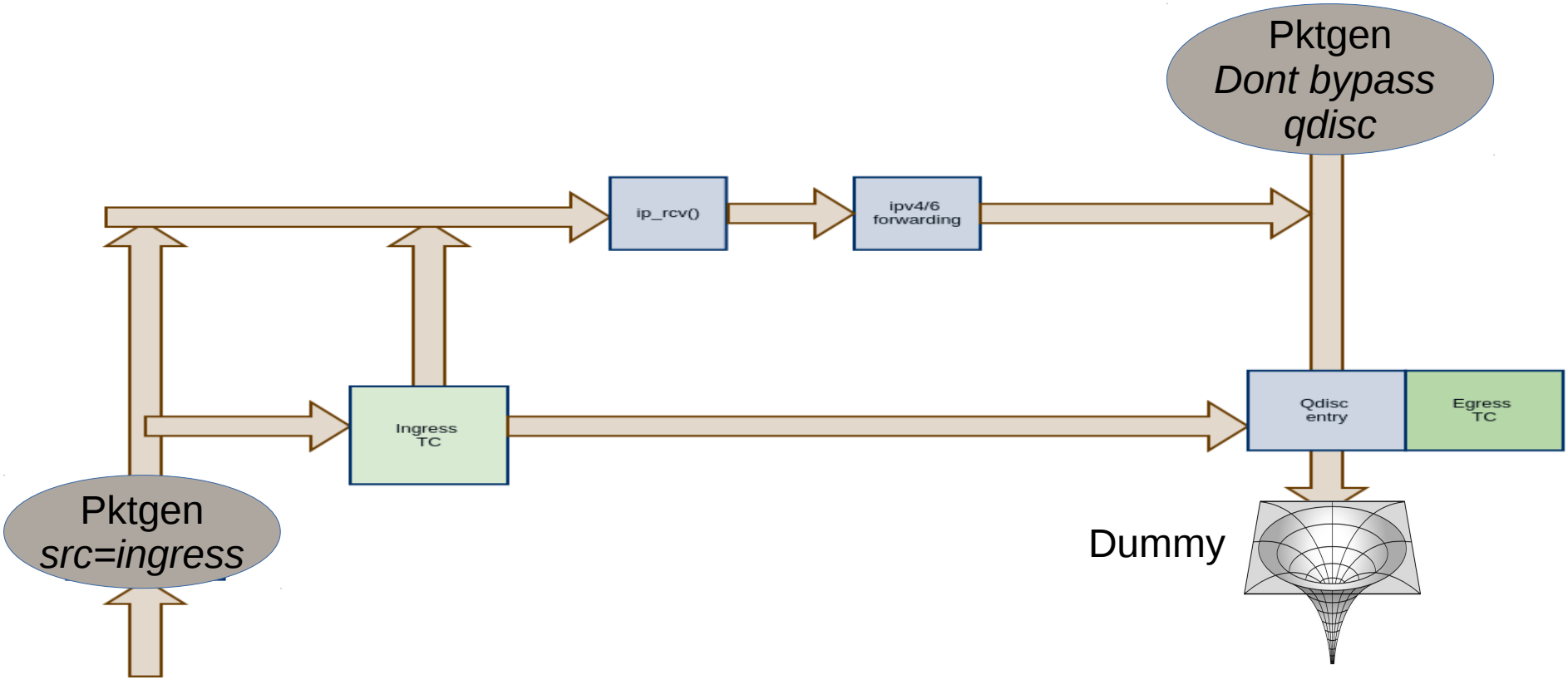


Focus On The SUT



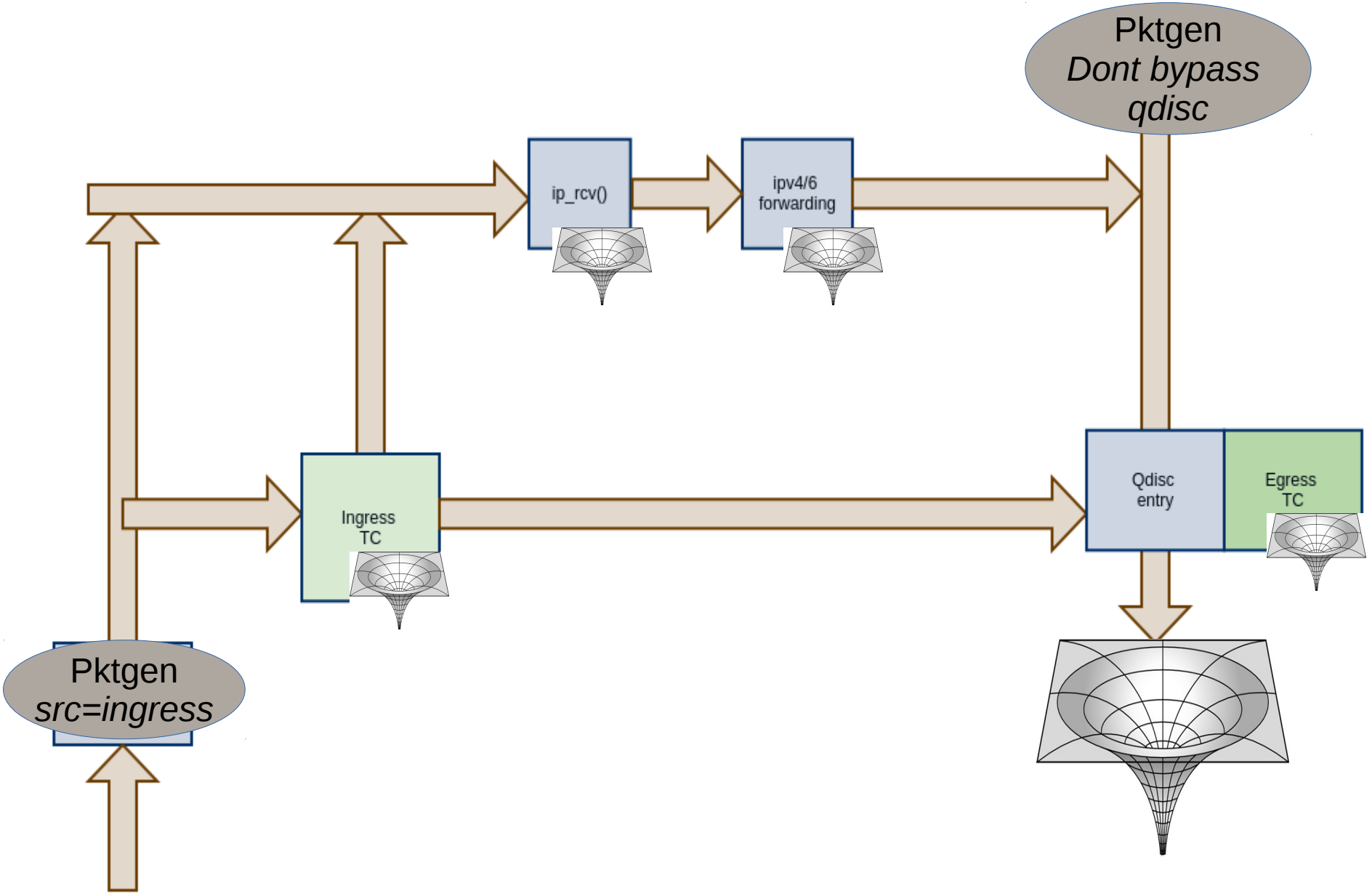
- Avoid system Locks and intermediate handoff queues
  - Use single core
  - Pktgen to source traffic from one CPU
    - Ingress sourcing by Alexei S.
    - Added Egress sourcing that didnt bypass qdisc

Focus On The SUT



- Get rid of Driver overhead
  - Use dummy netdev
    - Drops packets on the floor

Picking The Battle Scenes



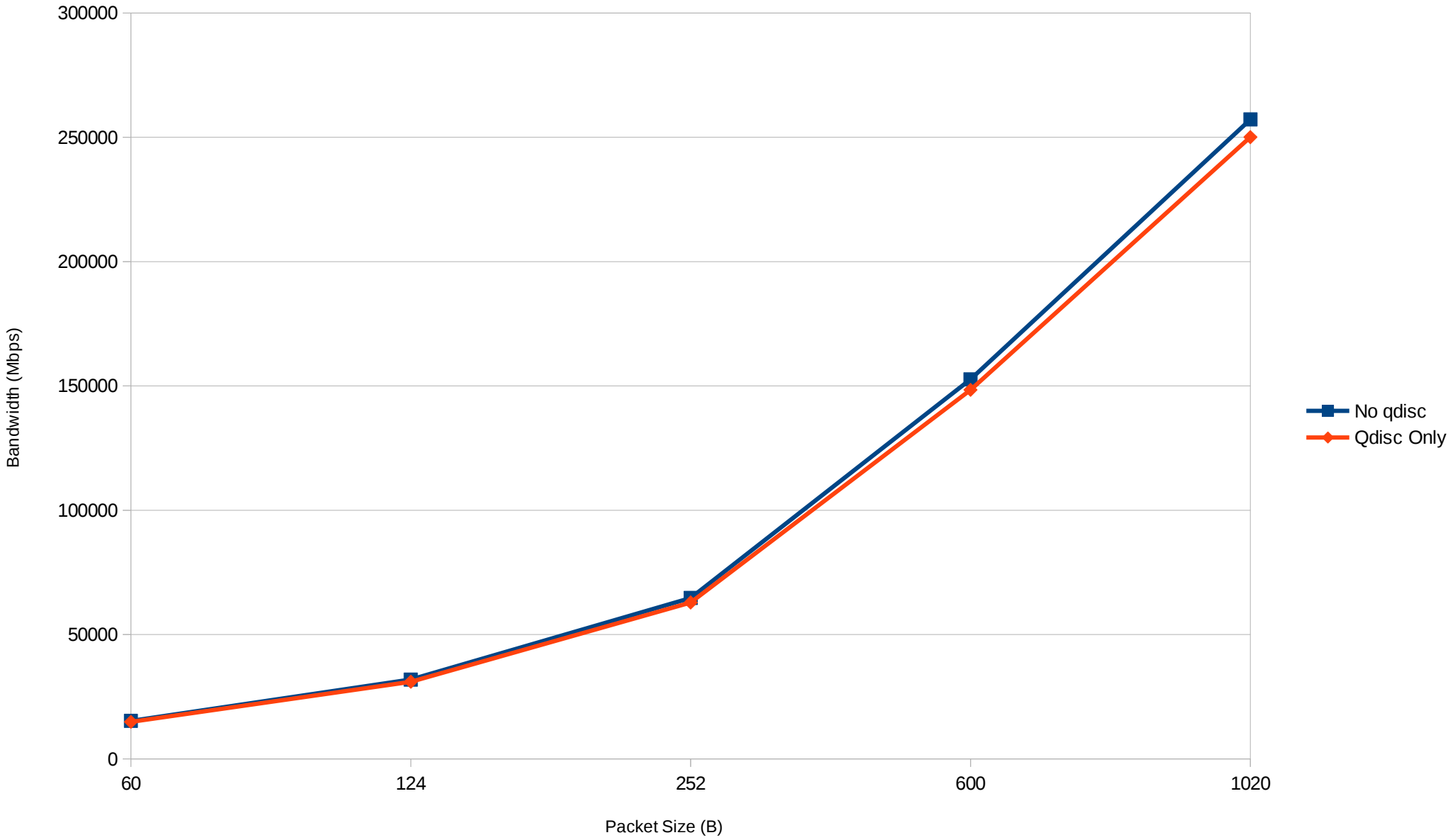
Focus On The SUT, kid

- Two weeks and a few thousand tests later...
  - Too many battle scenes!
  - We need to narrow it down to just the one scene
    - by the fountain beside city hall
      - And on a full moon at midnight
- Pick a hook to use
  - Baseline to justify it

Test Setup

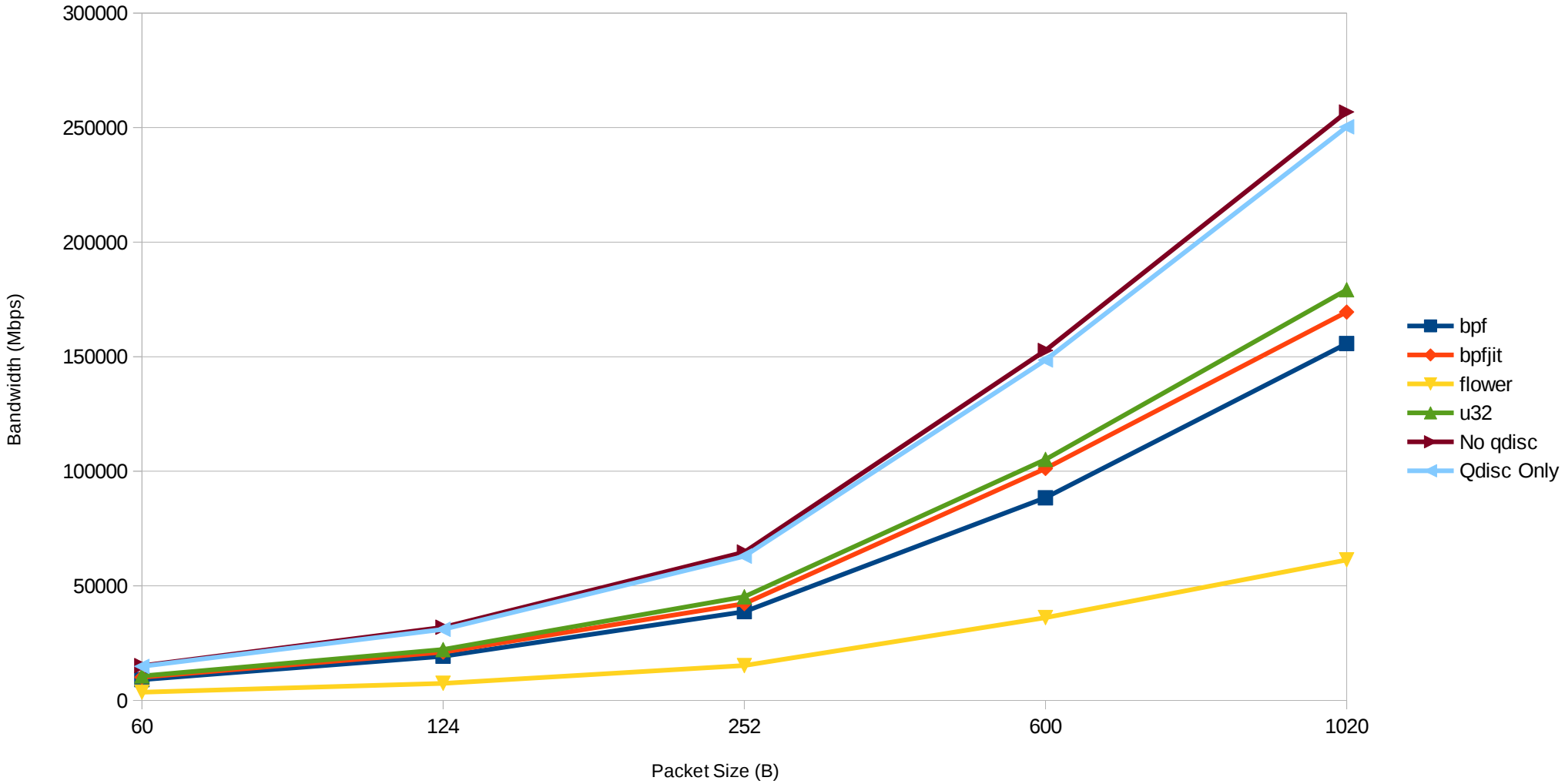
- Intel NUC
  - Quad Core i7-5557U CPU @ 3.10GHz
    - 800Mhz per cpu?
  - 16G 1600Mhz (dual stick) RAM
- Kernel
  - Net-next 4.4.1-rc1
    - Patched for flower and pktgen to do txmit qdisc

Baselining: Ingress Drop at *ip_rcv()*



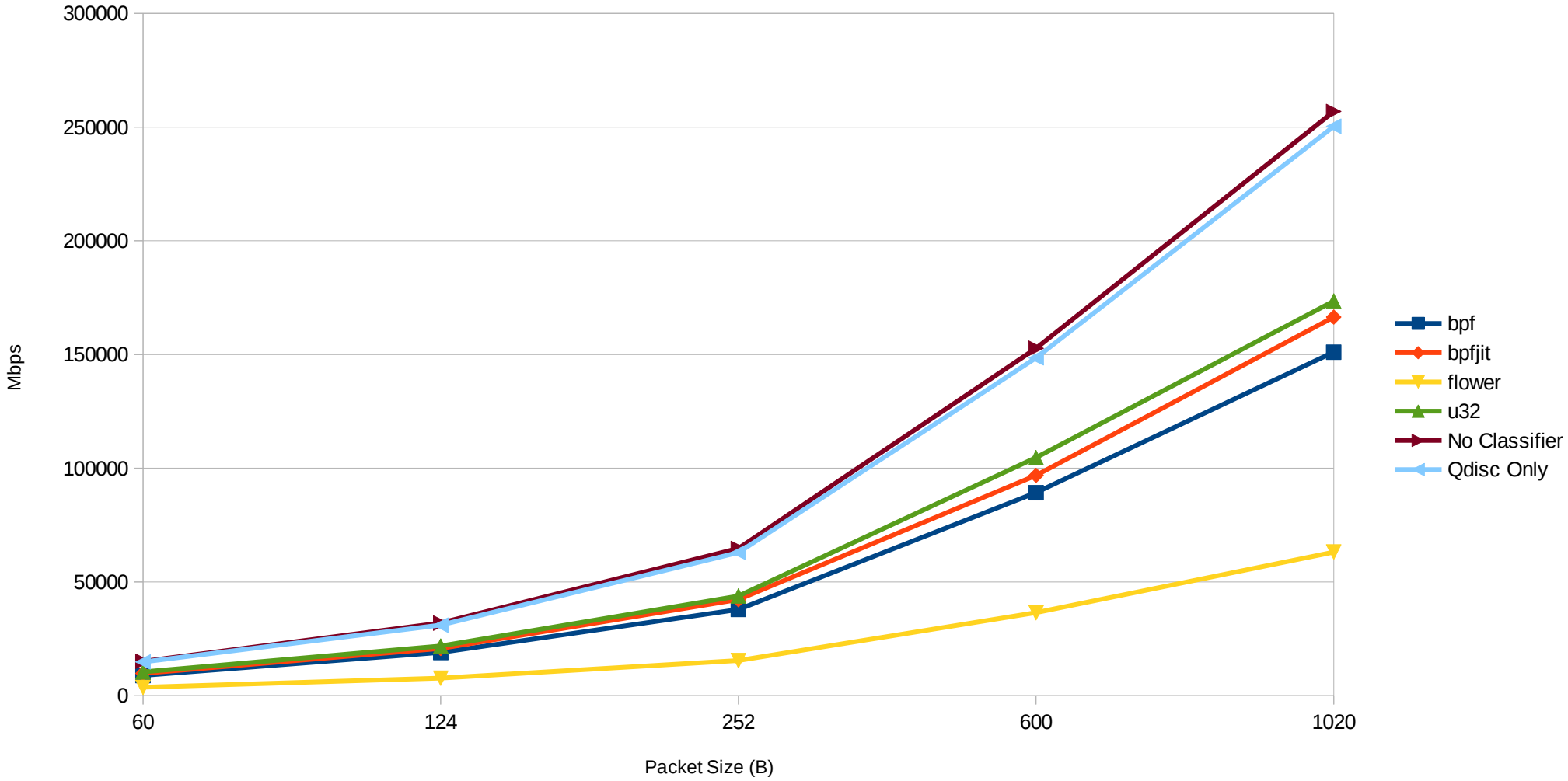
Baselining: Ingress Drop at *ip_rcv()*

- 1 rule(flow) with 1 tuple (src ip that matches)

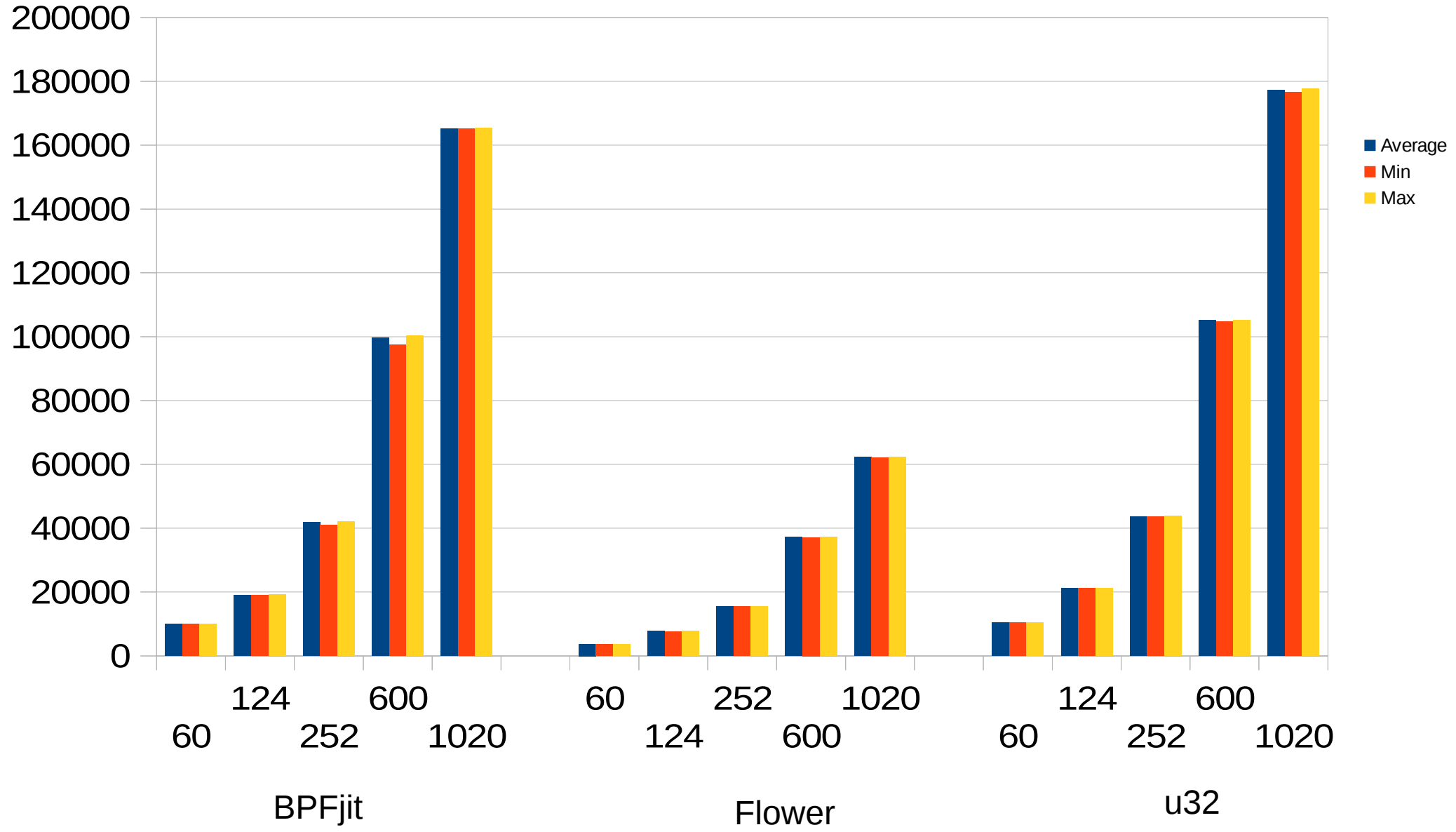


Baselining: Ingress Drop action

- 1 rule(flow) with 1 tuple (src ip) that matches



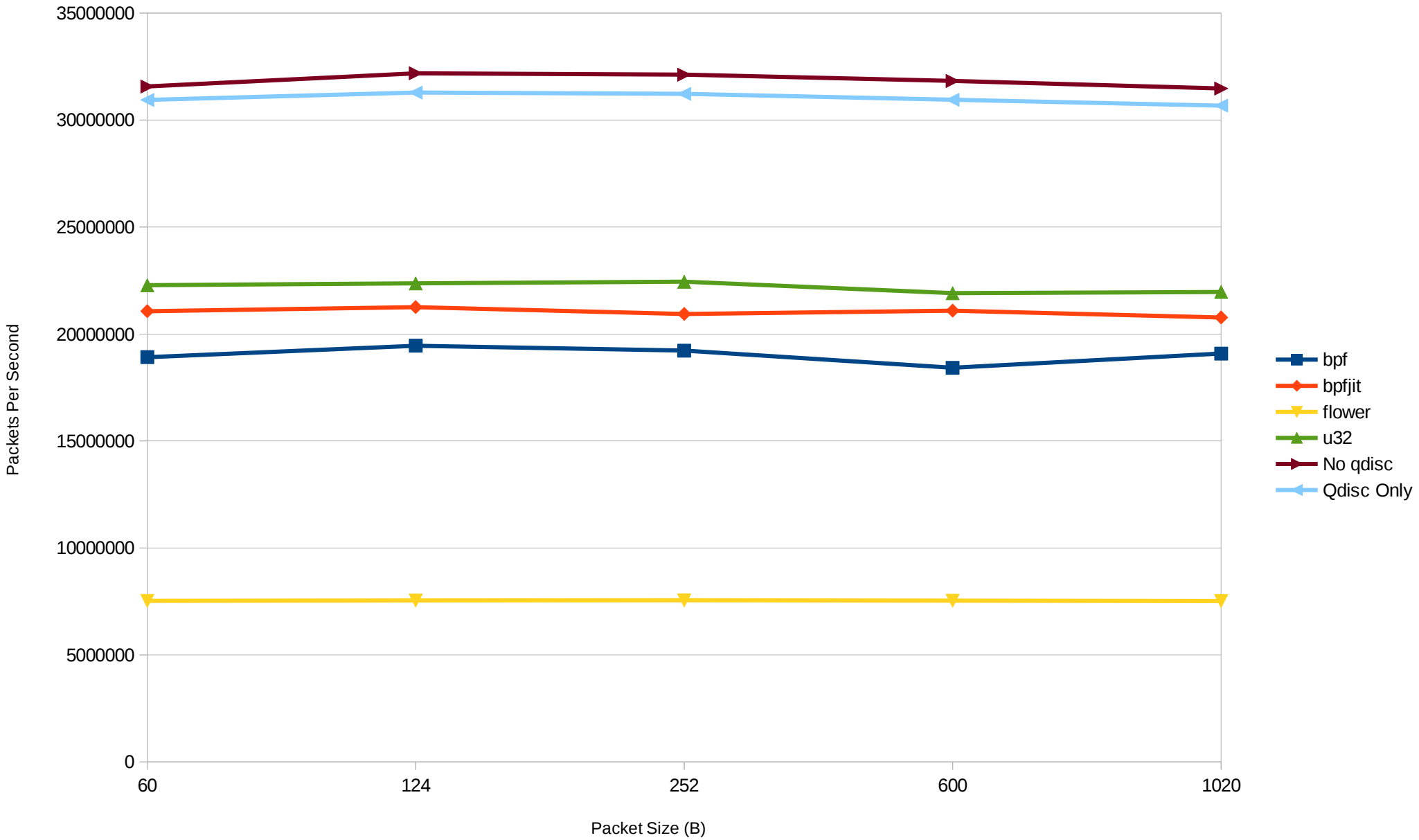
Baselining: Variance (mbps vs pktsize)



Cutting out tests

- Focus on average of 4 runs

Baselining Packet Size Effect



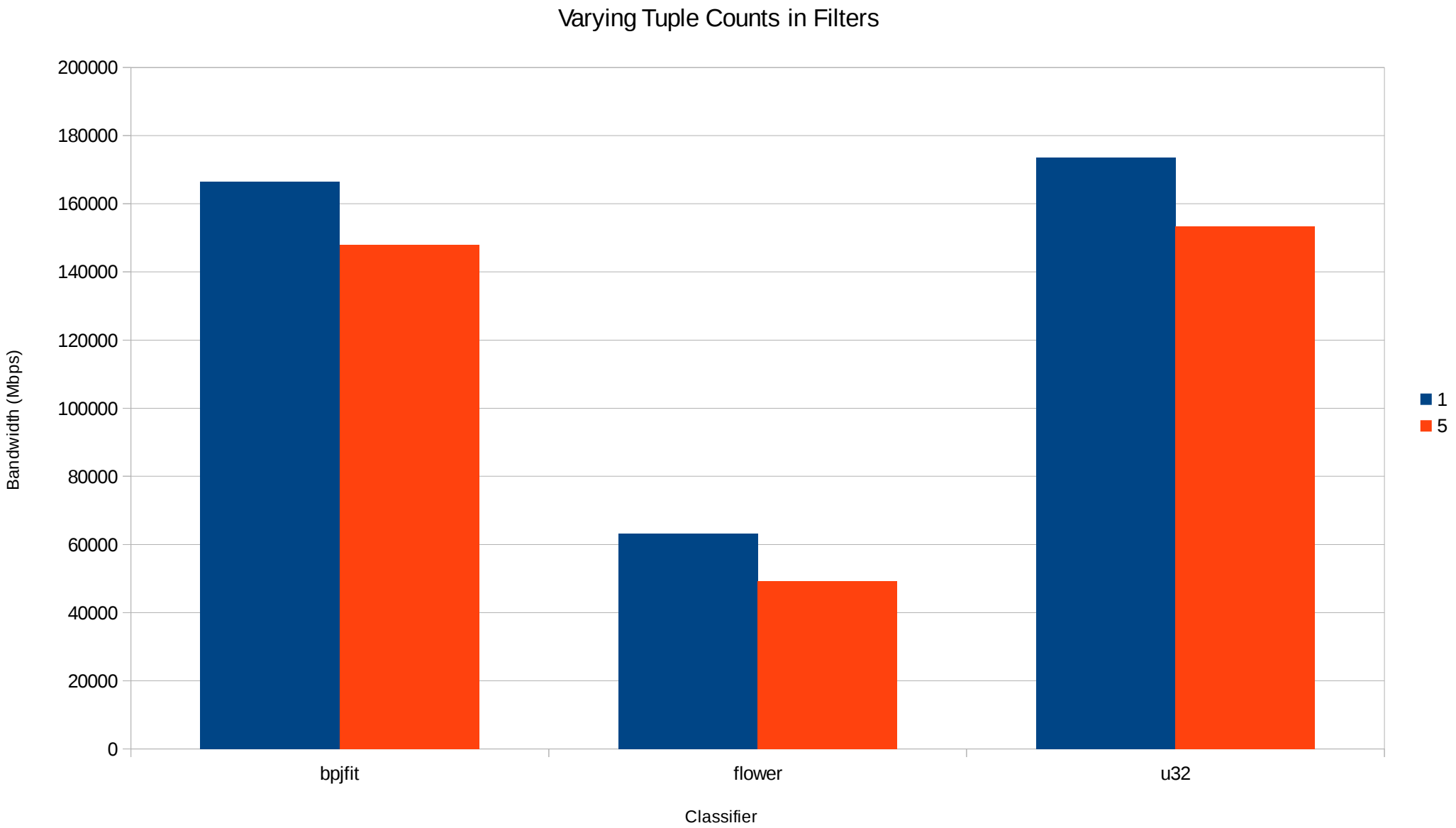
Cutting out tests

- Focus on average of 4 runs
- Use single packet size: 1020B

Cutting out tests

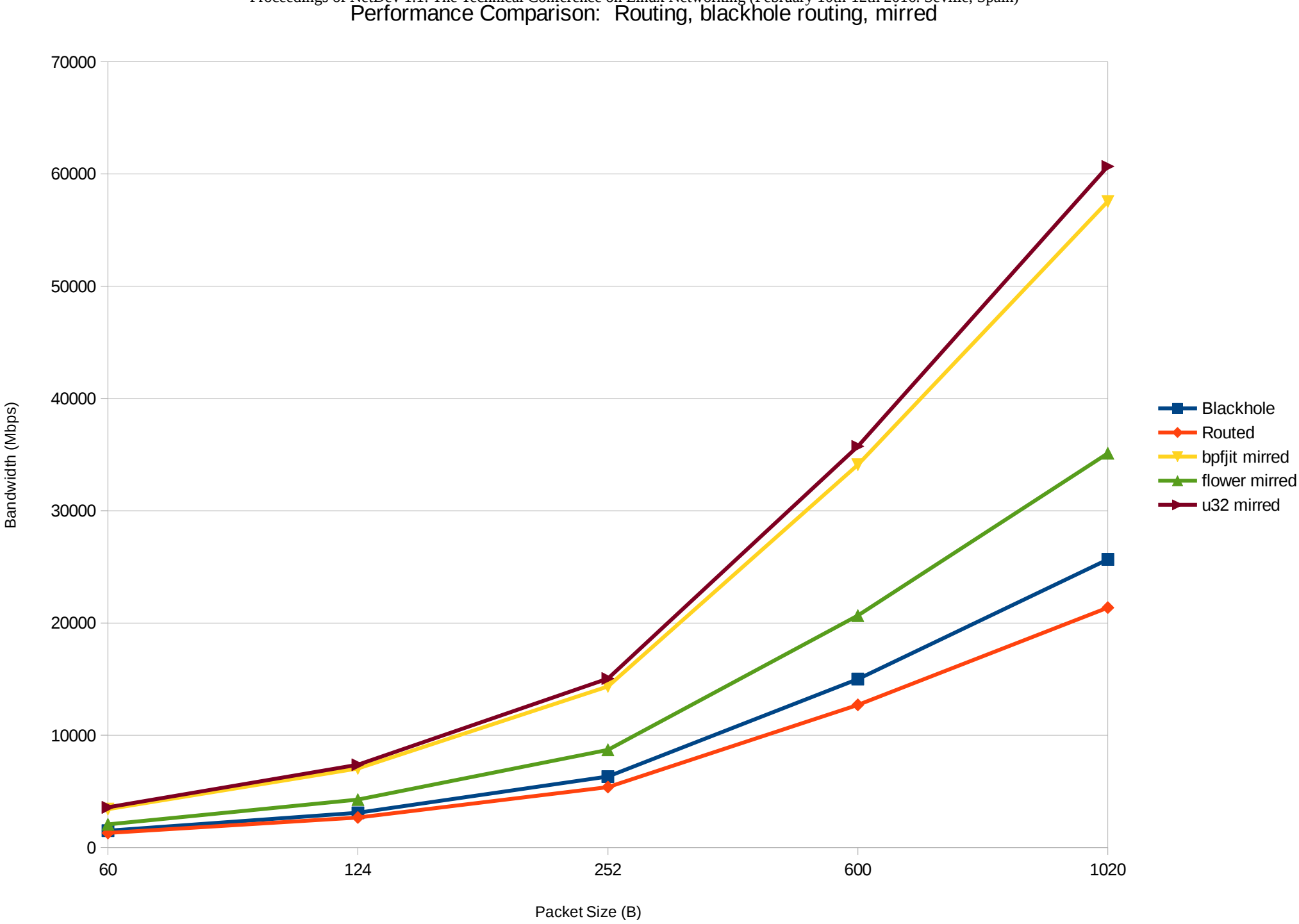
- Focus on average of 4 runs
- Use single packet size: 1020B
- Ignore bpf
  - All tests showing bpfjit doing better

Baselining: 1 vs 5 tuples(1020B)



Cutting out tests

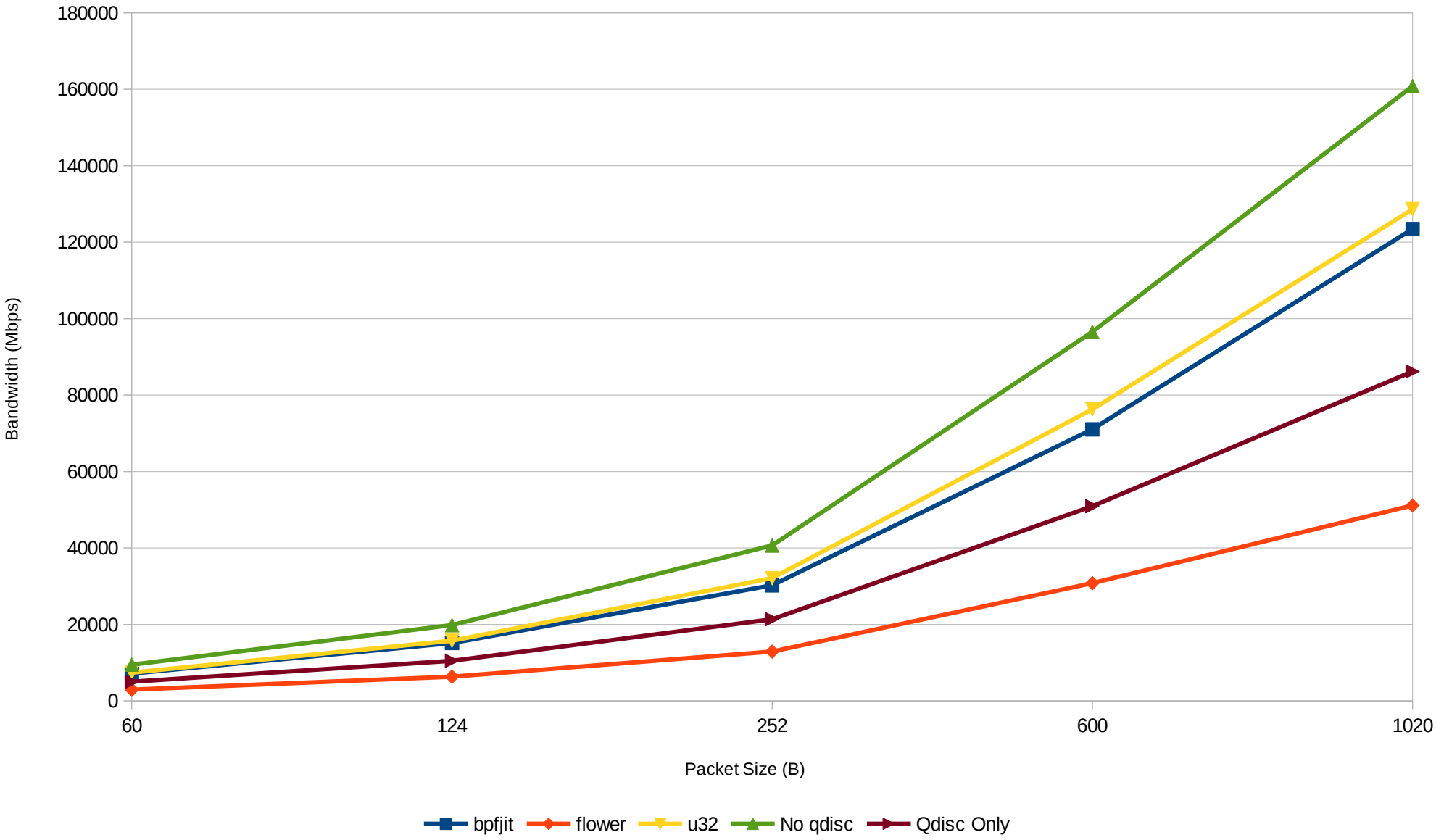
- Ignore bpf: Bpfjit is better
- Focus on average of 4 runs
- Use single packet size: 1020B
- Use 5 tuples because it is more realistic


Performance Comparison: Routing, blackhole routing, mirred
Bandwidth (Mbps)
70000
60000
50000
40000
30000
20000
10000
0
60
124
252
600
1020
Packet Size (B)
Blackhole
Routed
bpfjit mirred
flower mirred
u32 mirred

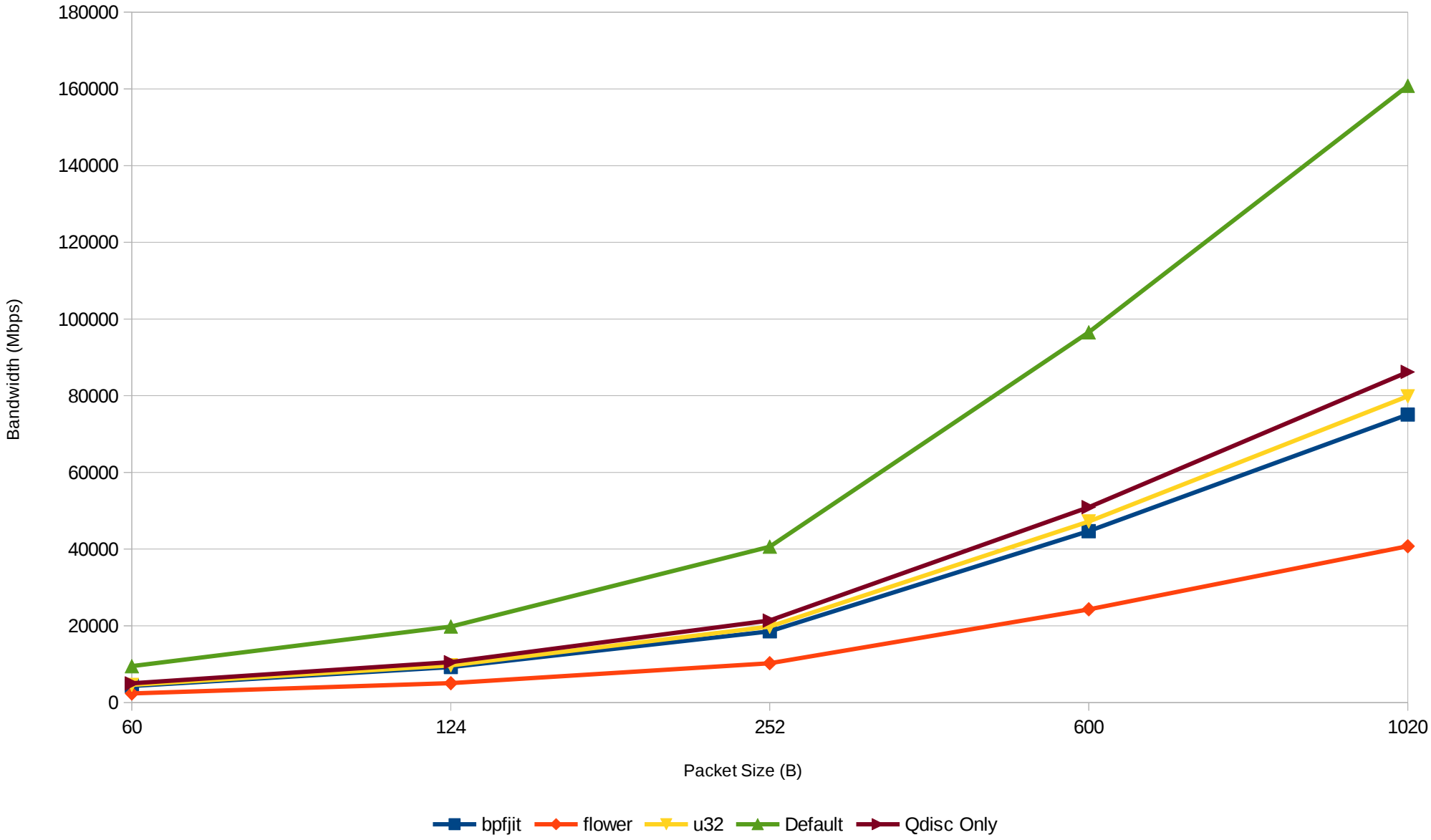
Cutting out tests

- Ignore bpf: Bpfjit is better
- Focus on average of 4 runs
- Use single packet size: 1020B
- Use 5 tuples because it is more realistic
- No forwarding tests because it contributes negatively

Baselining: Egress qdisc xmit Action: drop



Baselining: Egress qdisc xmit
No action



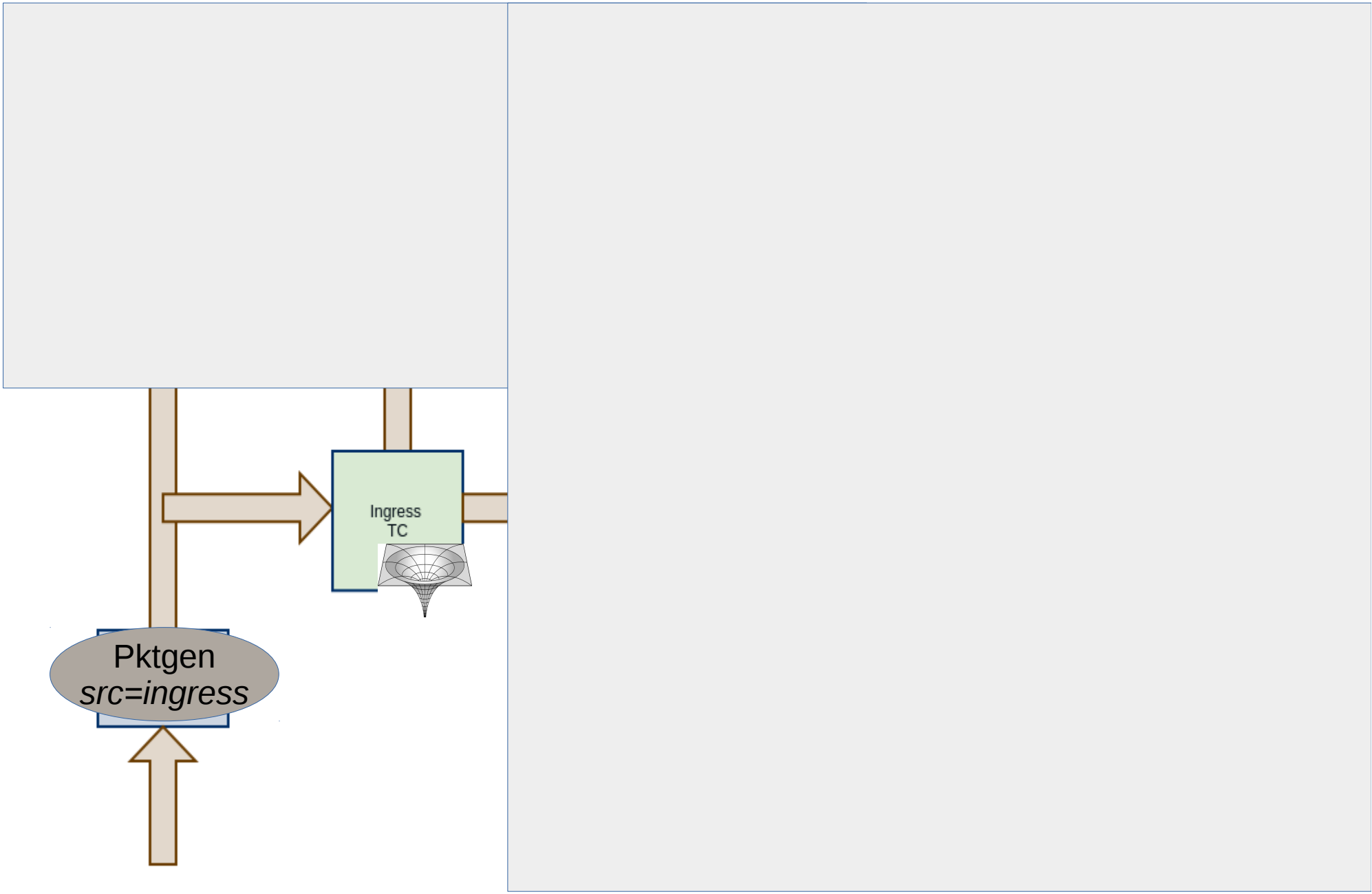
Cutting out tests

- Ignore bpf: Bpfjit is better
- Focus on average of 4 runs
- Use single packet size: 1020B
- Use 5 tuples because it is more realistic
- No forwarding tests because it will influence results
- No egress testing because it a lot slower
- Drop action so we can account for usage

Taking Stock

- The ingress can handle a lot of throughput
- The Forwarding code needs some investigation
- Memory latency is a bigger factor
  - Would like to buy a different latency RAM chips

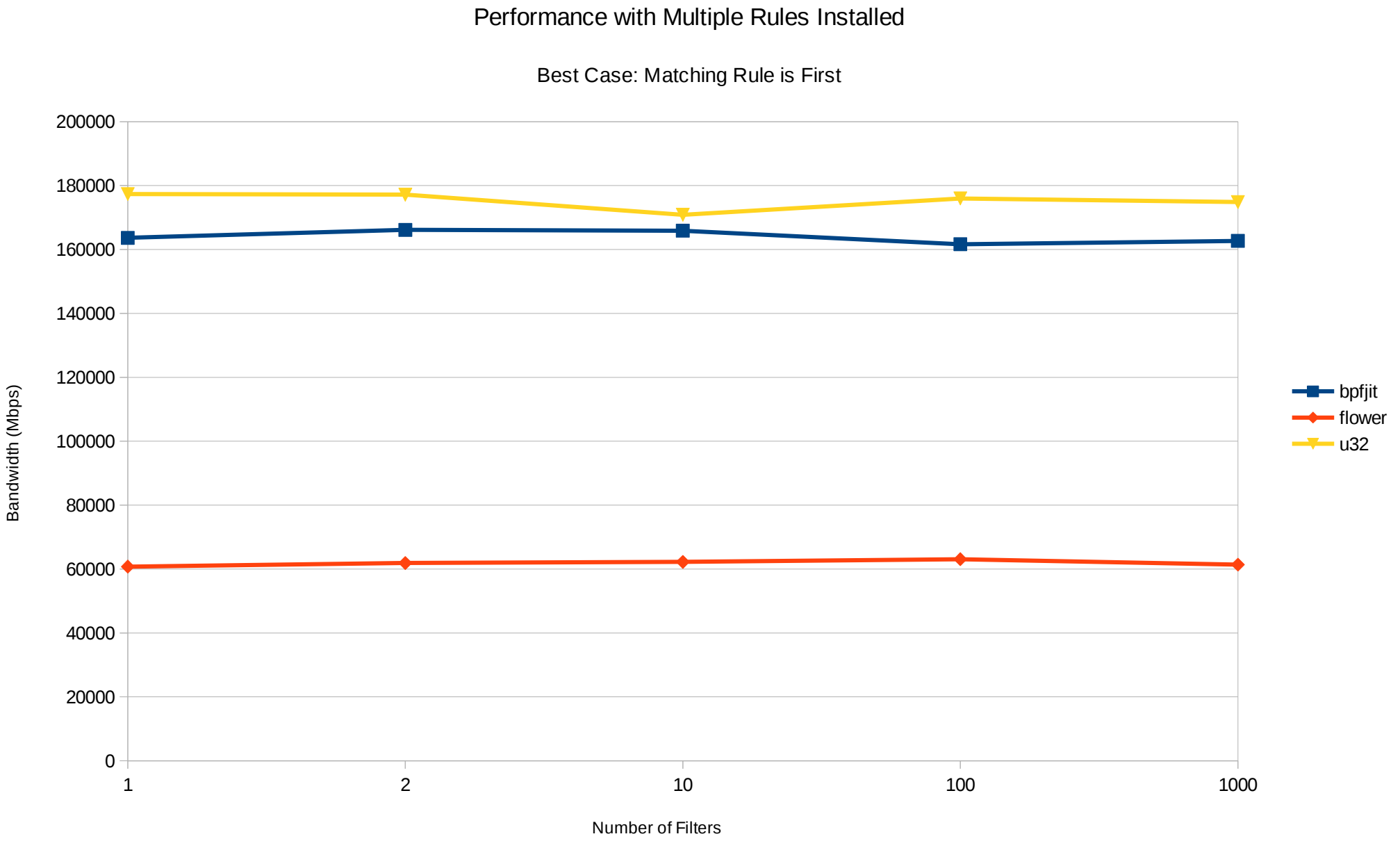
The Final Frontier



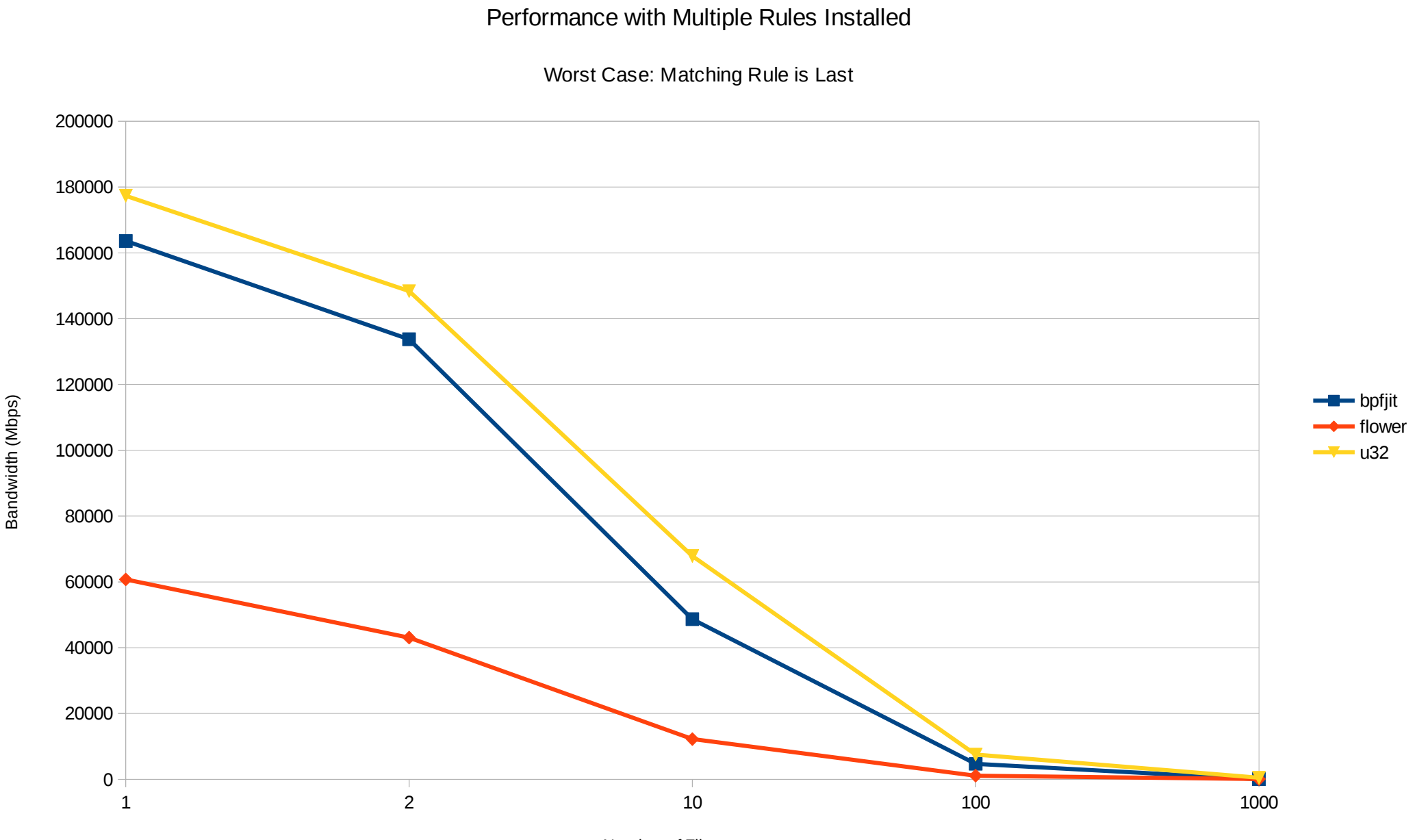
Preparing For Battle

- Battle Scene
  - Ingress qdisc
    - We know this is not the most fair test for flower
- Single cpu bursting from pktgen
- All Classifier filters have drop action
  - Provides accounting
  - Performance difference vs dropping at *ip_rcv* small
- 5 tuples
- 1020B packet size
- Batch of 4 tests each 25 secs

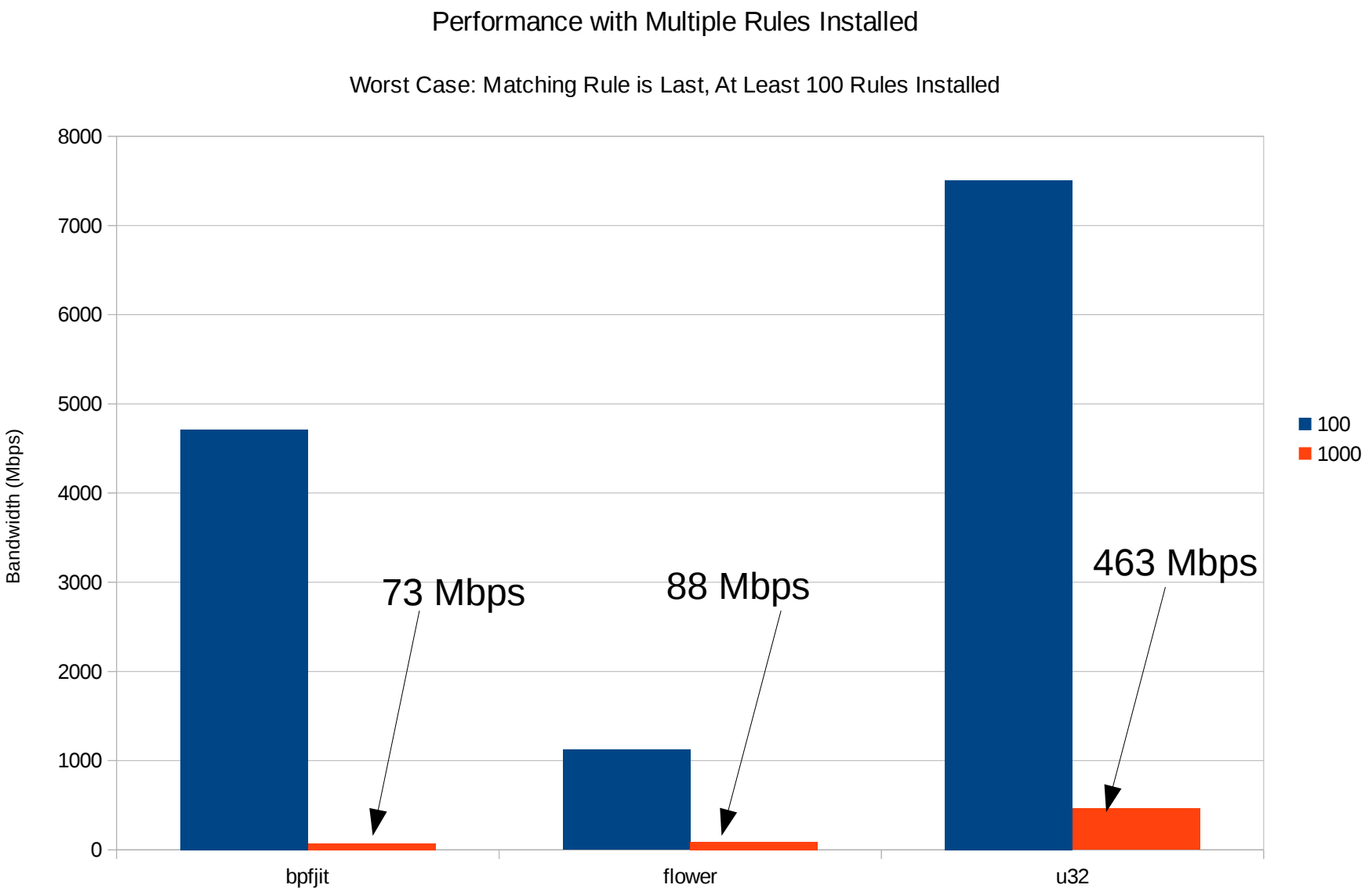
Let The Games Begin



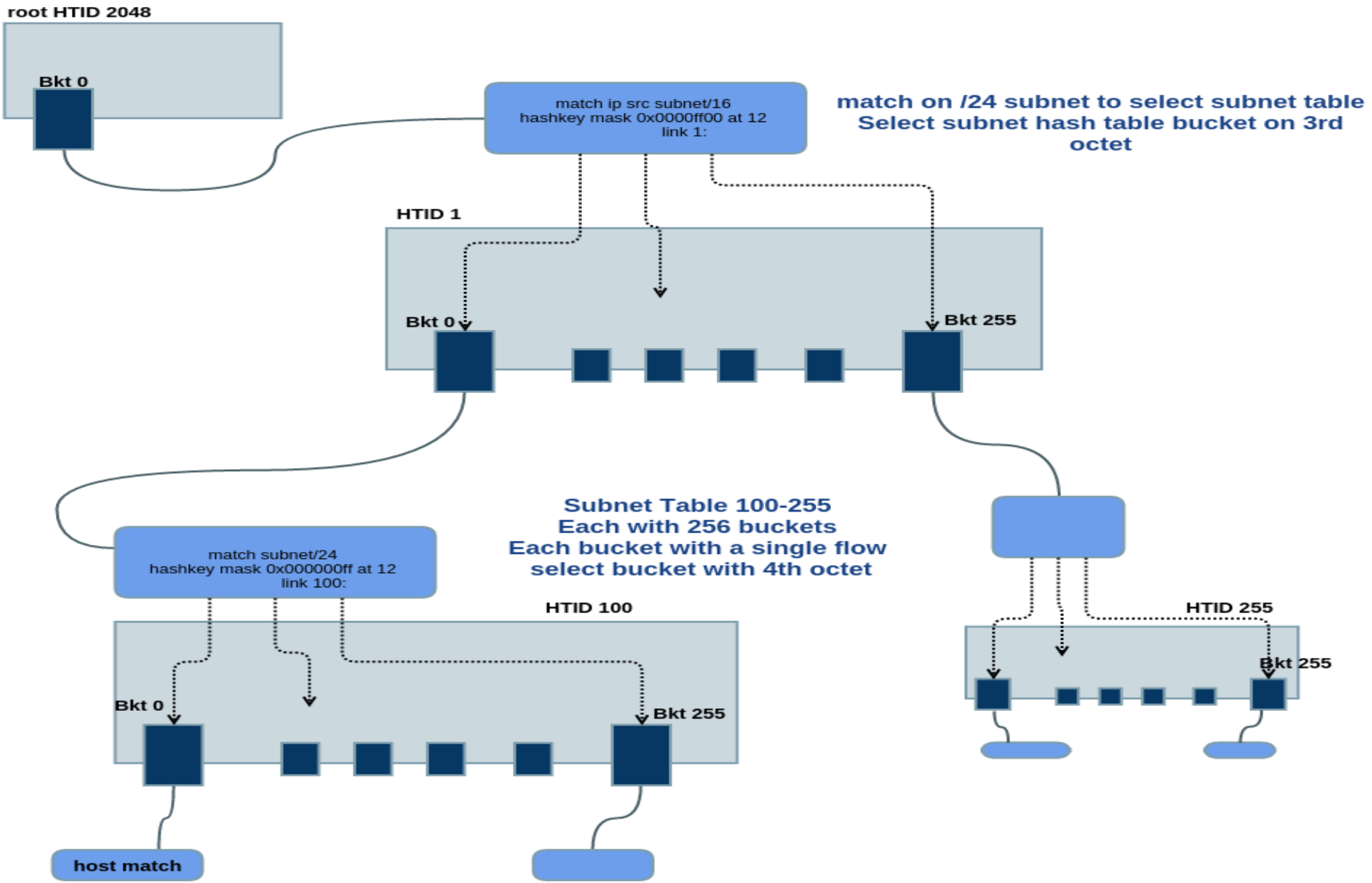
Let The Games Begin



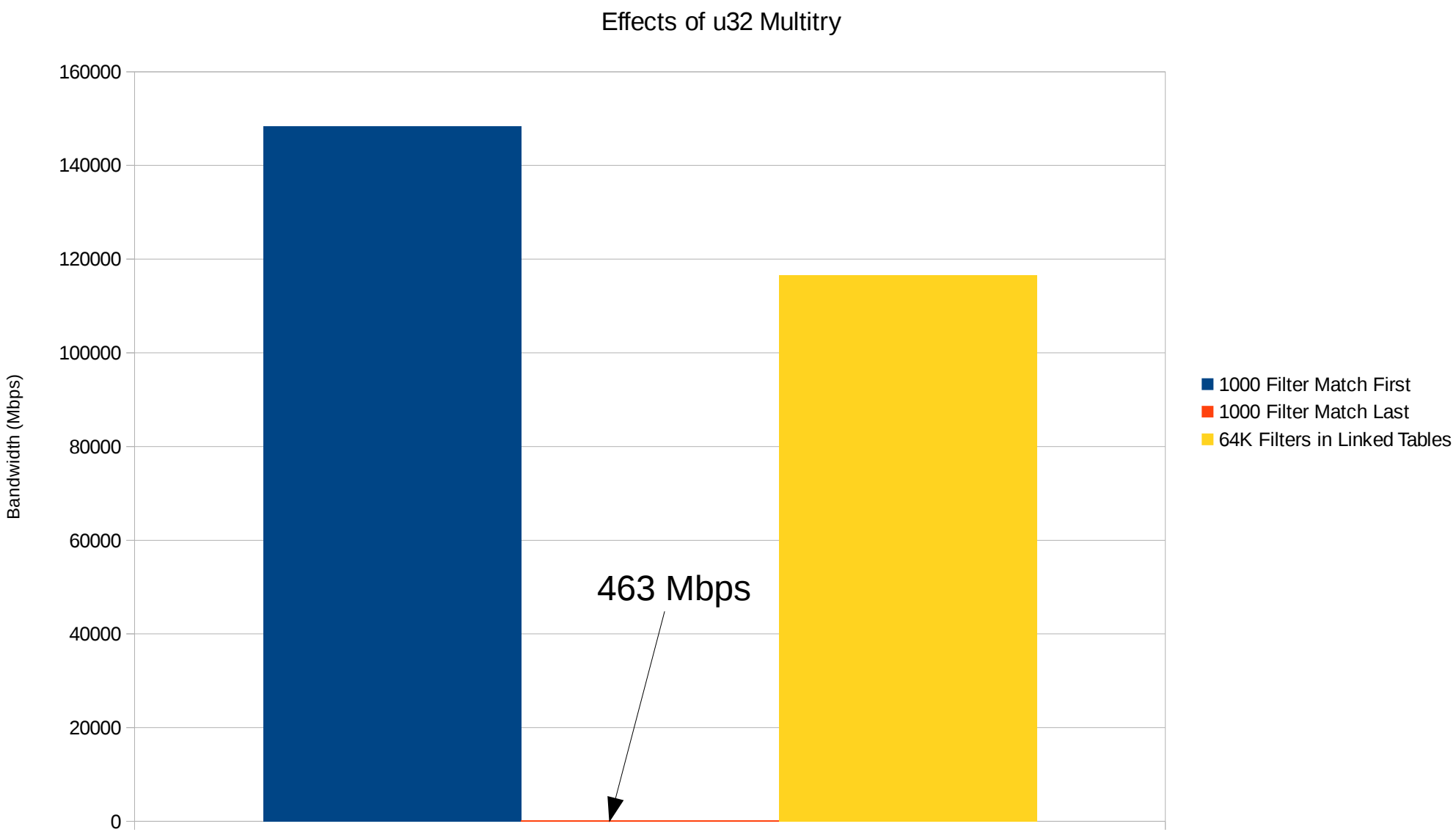
Let The Games Begin



U32 Multi-trie



U32 Multi-trie



Some Gotchas

- Lock Debugging is bad for performance

The Iron Triangle

- Performance
  - Data path processing
  - Control Path processing
- Usability
  - Admin level usability
- Extensibility
  - Via programmability or scriptability

References

- https://en.wikipedia.org/wiki/One_Ring
- "On getting tc classifier fully programmable with cls_bpf" (Daniel Borkmann) @netdev11
- “BPF In-kernel Virtual Machine Alexei Starovoitov” @netdev01
- “Implementing Open vSwitch datapath using TC” (Jiří Pírko) @netdev01
- “TC Classifier Action Subsystem Architecture” (Jamal Hadi Salim) @netdev01
- “clsact: kernel commit 1f211a1b929c804100e138c5d3d656992cfd5622”

  (Daniel Borkmann)

Diagrams

- http://www.dummies.com/how-to/content/string-theory-defining-a-black-hole.html
-