Flexible Control of Parallelism in a Multiprocessor PC Router

Benjie Chen and Robert Morris

MIT LCS

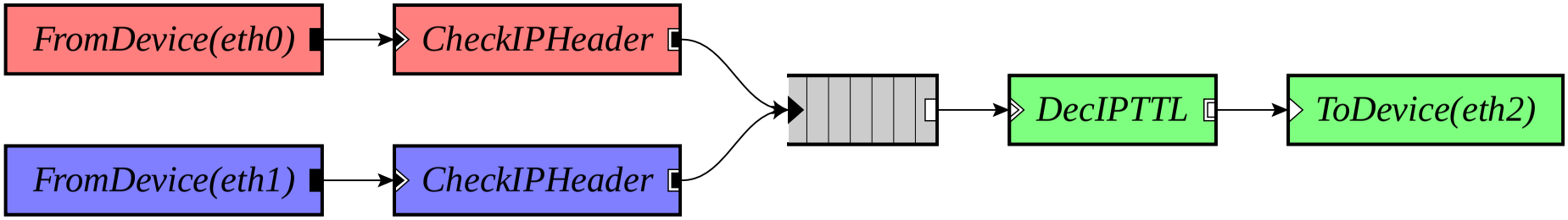
http://www.pdos.lcs.mit.edu/click/

Motivation

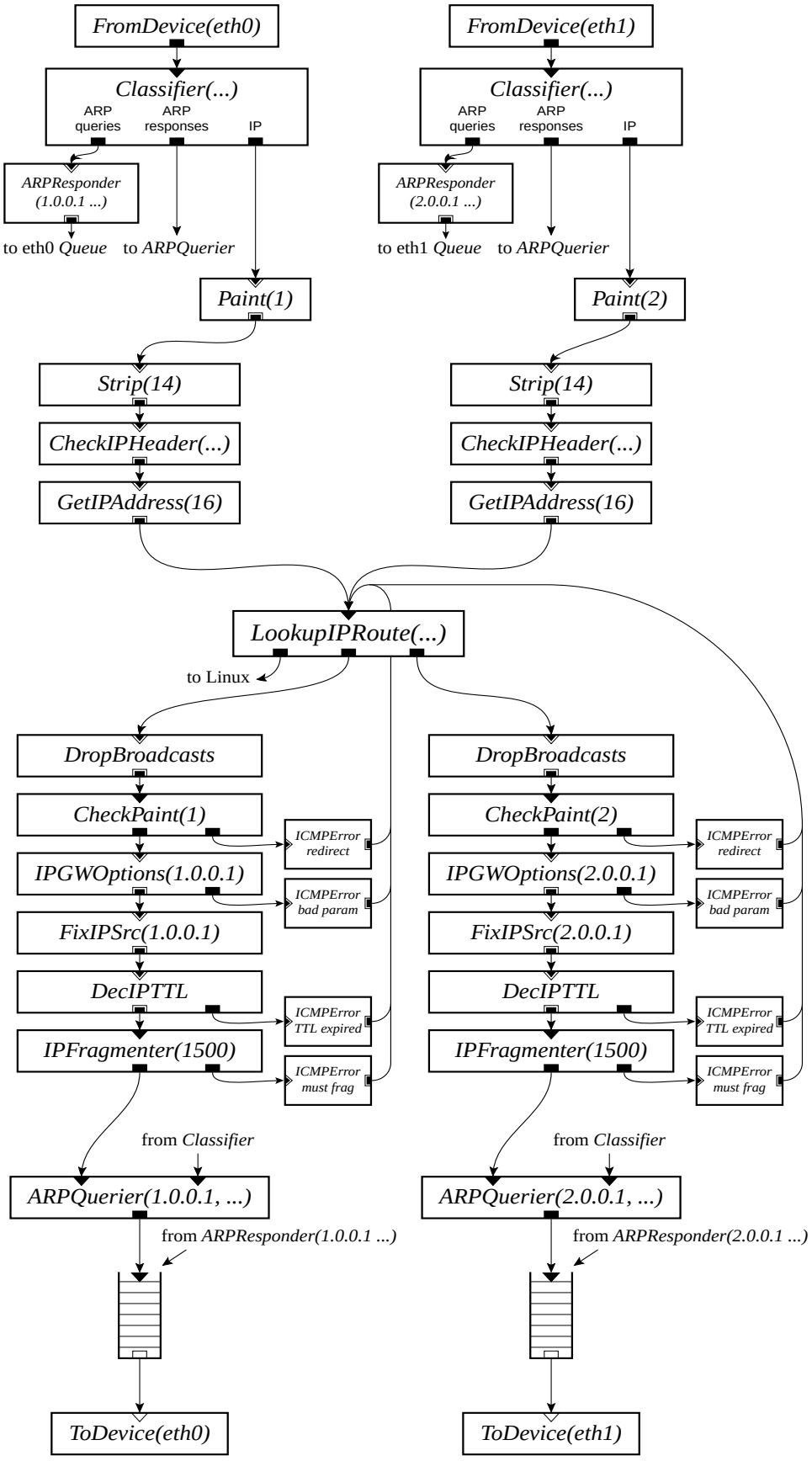
- Click eases software router construction
- Many packet processing tasks require more CPU cycles

  Encryption, NAT, Firewall
- Click eases construction of parallel packet processing tasks

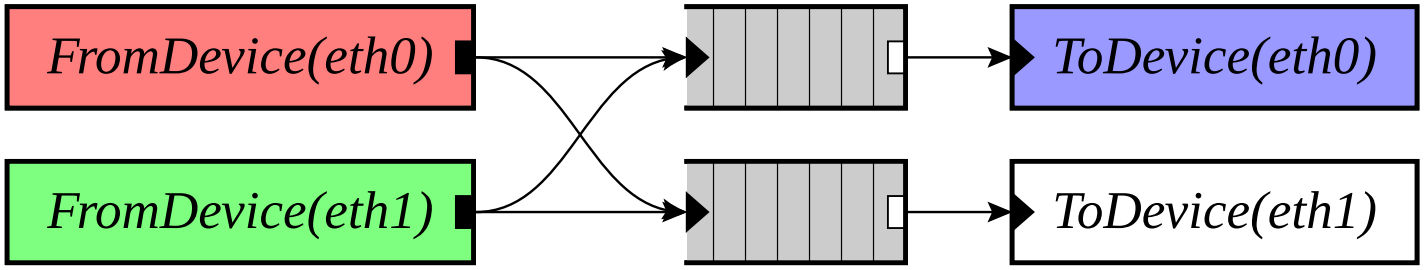
Click



- A router is a collection of interconnected *Elements*
- Packet enters configuration at `FromDevice`
- Packet moves using function calls
- Packet stops at `Queue` or `ToDevice`

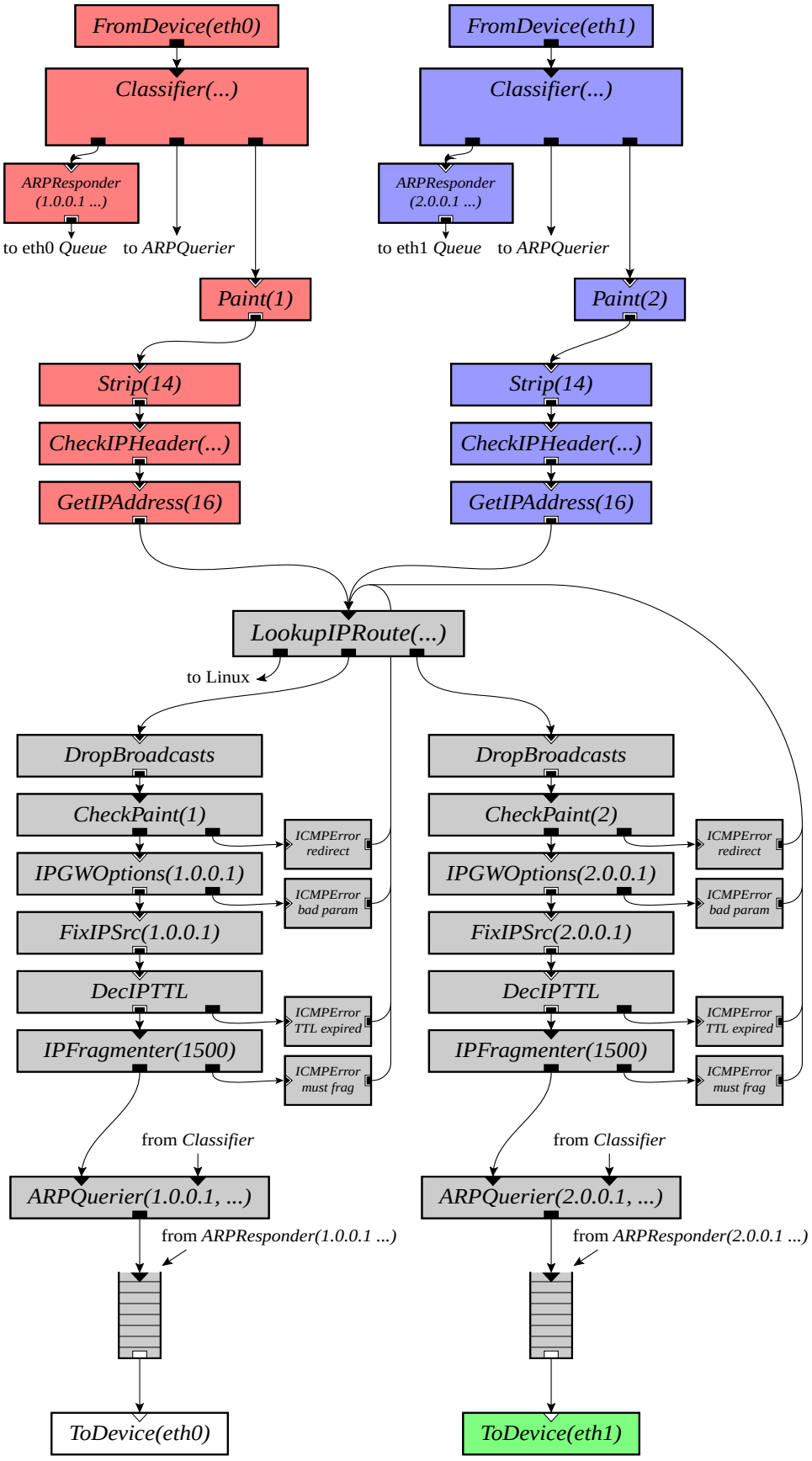

FromDevice(eth0)
Classifier(...)
ARP queries
ARP responses
IP
ARPResponder (1.0.0.1 ...)
to eth0 Queue
to ARPQuerier
Paint(1)
Strip(14)
CheckIPHeader(...)
GetIPAddress(16)
FromDevice(eth1)
Classifier(...)
ARP queries
ARP responses
IP
ARPResponder (2.0.0.1 ...)
to eth1 Queue
to ARPQuerier
Paint(2)
Strip(14)
CheckIPHeader(...)
GetIPAddress(16)
LookupIPRoute(...)
to Linux
DropBroadcasts
CheckPaint(1)
ICMPError redirect
IPGWOptions(1.0.0.1)
ICMPError bad param
FixIPSrc(1.0.0.1)
DecIPTTL
ICMPError TTL expired
IPFragmenter(1500)
ICMPError must frag
from Classifier
ARPQuerier(1.0.0.1, ...)
from ARPResponder(1.0.0.1 ...)
ToDevice(eth0)
DropBroadcasts
CheckPaint(2)
ICMPError redirect
IPGWOptions(2.0.0.1)
ICMPError bad param
FixIPSrc(2.0.0.1)
DecIPTTL
ICMPError TTL expired
IPFragmenter(1500)
ICMPError must frag
from Classifier
ARPQuerier(2.0.0.1, ...)
from ARPResponder(2.0.0.1 ...)
ToDevice(eth1)

Configuration Exhibits Natural Parallelism



- Each `FromDevice` or `ToDevice` can run on a separate CPU
- `Queue` hands packets from one CPU to another

  `Queue` needs to be SMP safe


FromDevice(eth0)
Classifier(...)
ARPResponder (1.0.0.1 ...)
to eth0 Queue
to ARPQuerier
Paint(1)
Strip(14)
CheckIPHeader(...)
GetIPAddress(16)
FromDevice(eth1)
Classifier(...)
ARPResponder (2.0.0.1 ...)
to eth1 Queue
to ARPQuerier
Paint(2)
Strip(14)
CheckIPHeader(...)
GetIPAddress(16)
LookupIPRoute(...)
to Linux
DropBroadcasts
CheckPaint(1)
ICMPError redirect
IPGWOptions(1.0.0.1)
ICMPError bad param
FixIPSrc(1.0.0.1)
DecIPTTL
ICMPError TTL expired
IPFragmenter(1500)
ICMPError must frag
from Classifier
ARPQuerier(1.0.0.1, ...)
from ARPResponder(1.0.0.1 ...)
ToDevice(eth0)
DropBroadcasts
CheckPaint(2)
ICMPError redirect
IPGWOptions(2.0.0.1)
ICMPError bad param
FixIPSrc(2.0.0.1)
DecIPTTL
ICMPError TTL expired
IPFragmenter(1500)
ICMPError must frag
from Classifier
ARPQuerier(2.0.0.1, ...)
from ARPResponder(2.0.0.1 ...)
ToDevice(eth1)

SMP Click Goals

- Uniprocessor configurations should work correctly

  User need not worry about race conditions

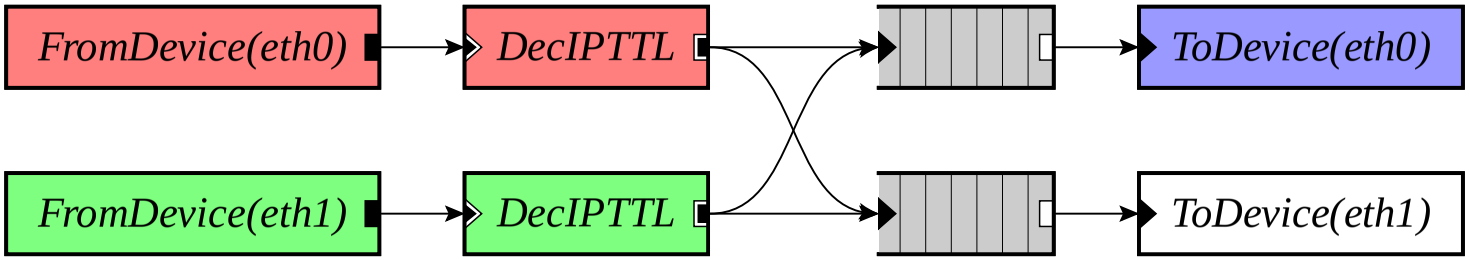
- Take advantage of natural parallelism in configurations

- Expose more parallelism with simple changes

Implementation

- `FromDevice` and `ToDevice` elements run on separate CPUs
- Synchronized `Queue` transfers packets between CPUs
- Adaptive scheduler maps `FromDevice` and `ToDevice` onto CPUs

Scheduling

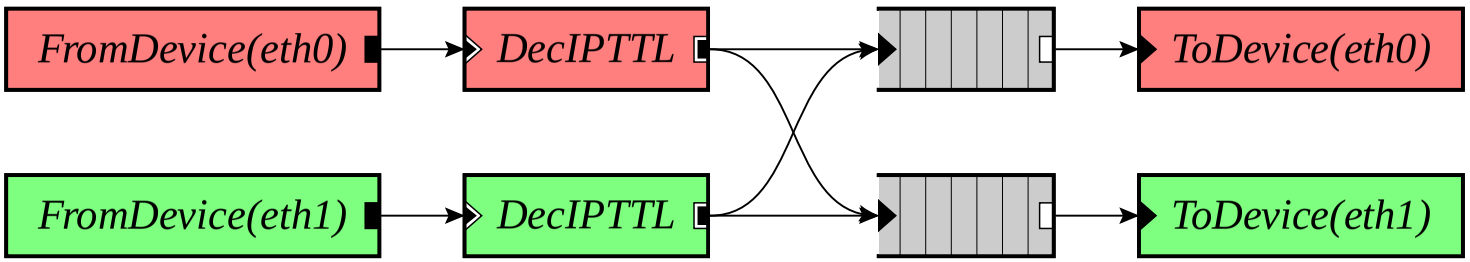


- Work initiated by `FromDevice` elements are more expensive
- Adaptive scheduler balances load across CPUs

Adaptive Load Balancing

- SMP Click samples execution time of elements
- Periodically (e.g. 1 second), runs bin packing algorithm:
  - Sorts all schedulable elements based on cost
  - Assigns element to thread with smallest total cost so far
- Benefits
  - Provide good load balance
  - Adapts to traffic pattern

Limitations of the Adaptive Approach



- Cannot statically determine common path from configuration
- Some assignments trigger more cache misses
- SMP Click allows programmer to specify assignment

Implementation is Simple

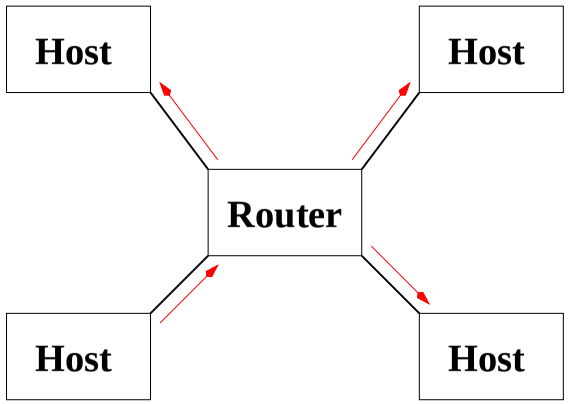
- 19 out of 161 elements need to be changed

  14 out of 19 only needed atomic `incr` for counters

- Adaptive scheduler: 290 lines of C++ code and comments

- Implementation and performance tuning took 3 months

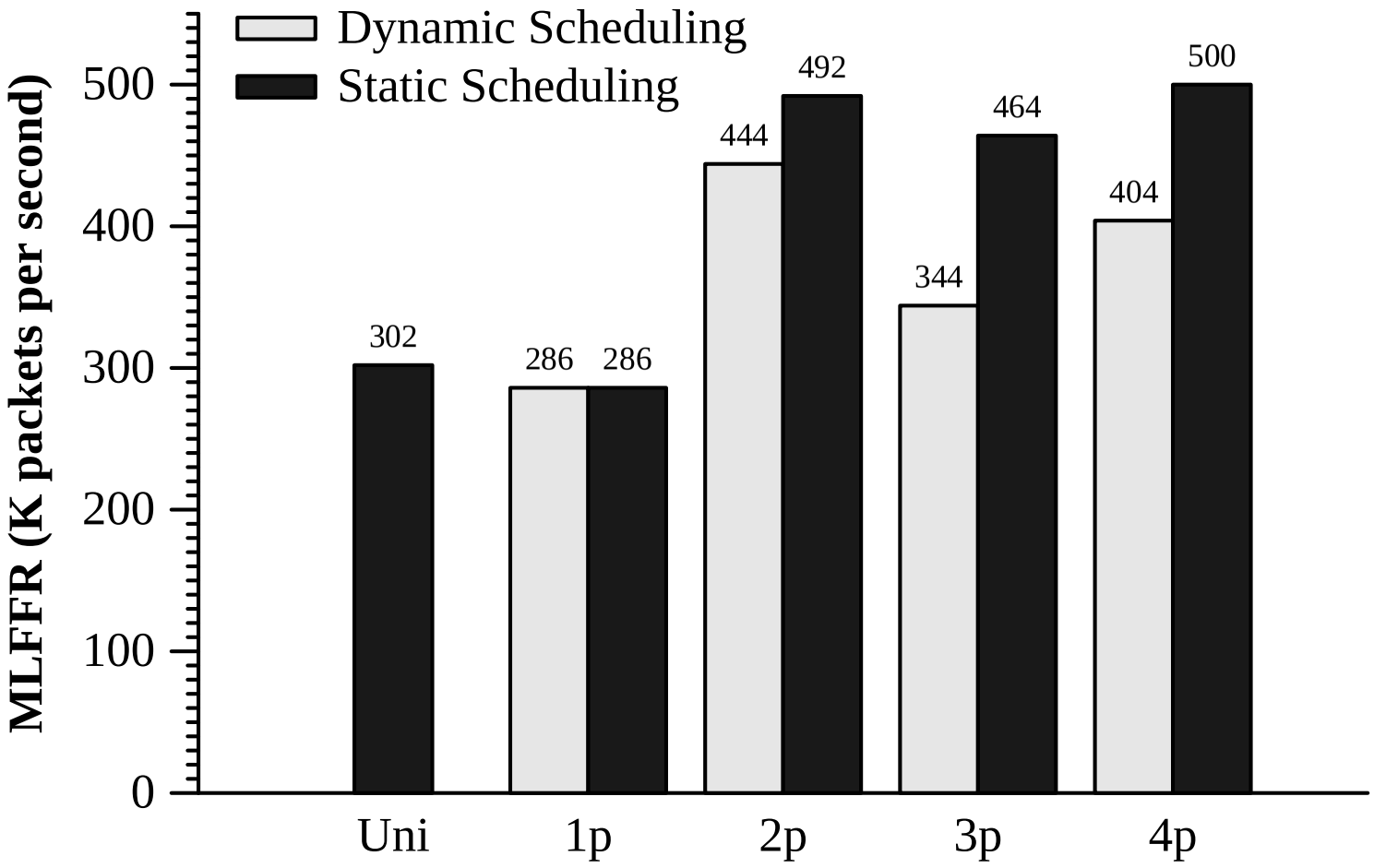
Experimental Setup



- Each host sends 64 byte packets to 3 other hosts

  Total of 12 streams

IP Router Performance



IP Router Does Not Scale Well

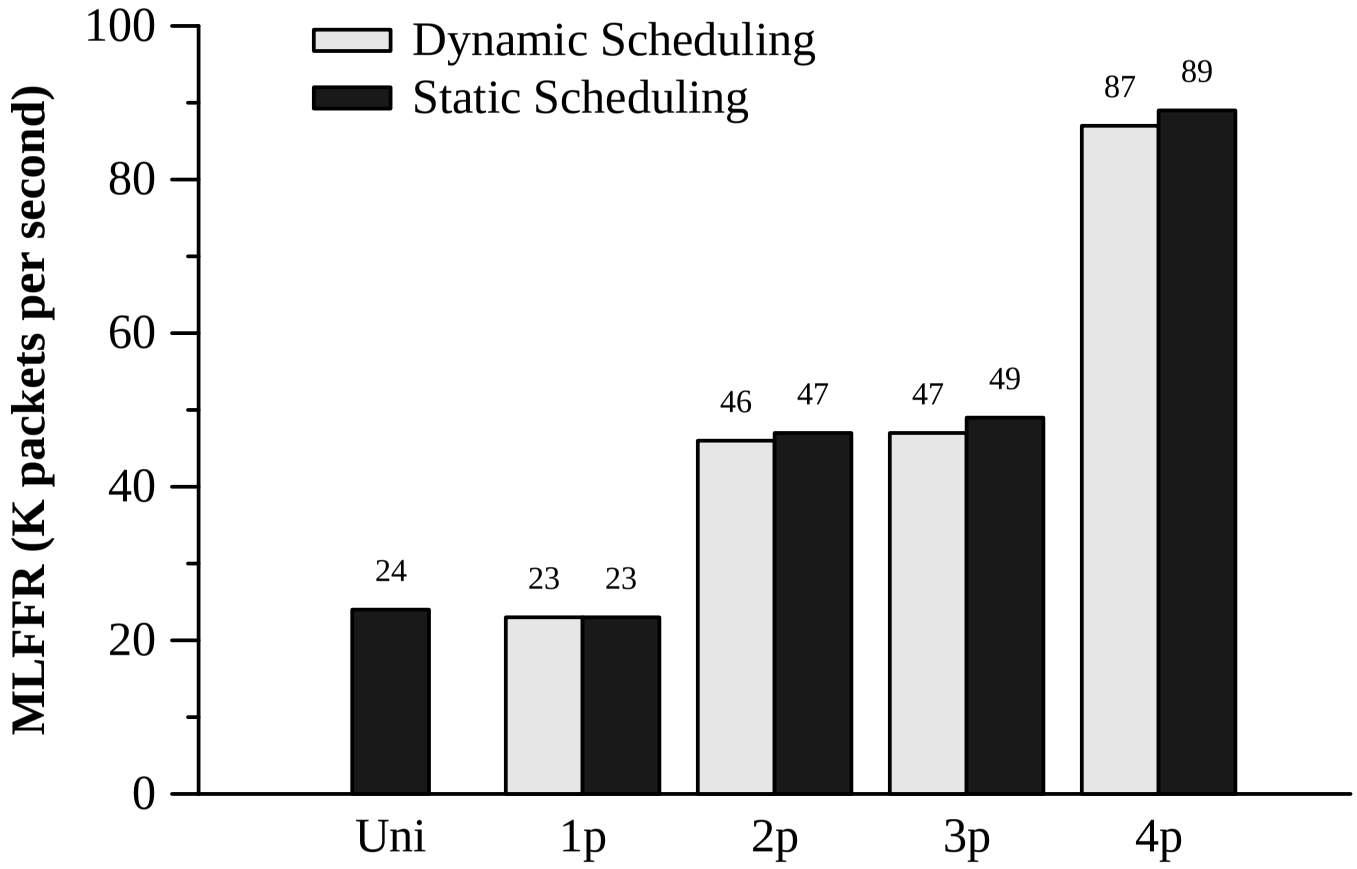
- Cost of enqueue increases due to lock contention
- Number of cache misses increases

  On 1 CPU, SMP Click forwards each packet in 3.5 μs

  Cost of moving a packet between two CPUs is 3.0 μs

- SMP Click does not payoff when cost of computation on each packet is near the cost of cache misses

IPsec Performance



- SMP Click pays off when packet processing cost >> cost of cache misses

SMP Click Usefulness

- Speed up is limited by cost of cache misses

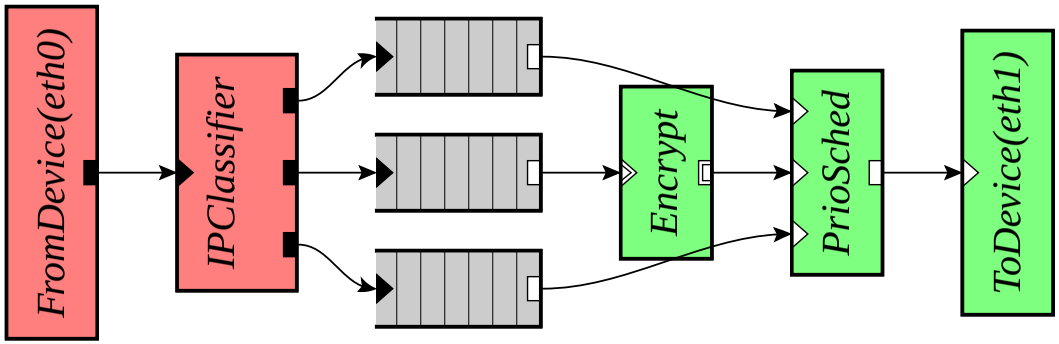
  e.g. IP router

- Speed up improves as cost of computation increases

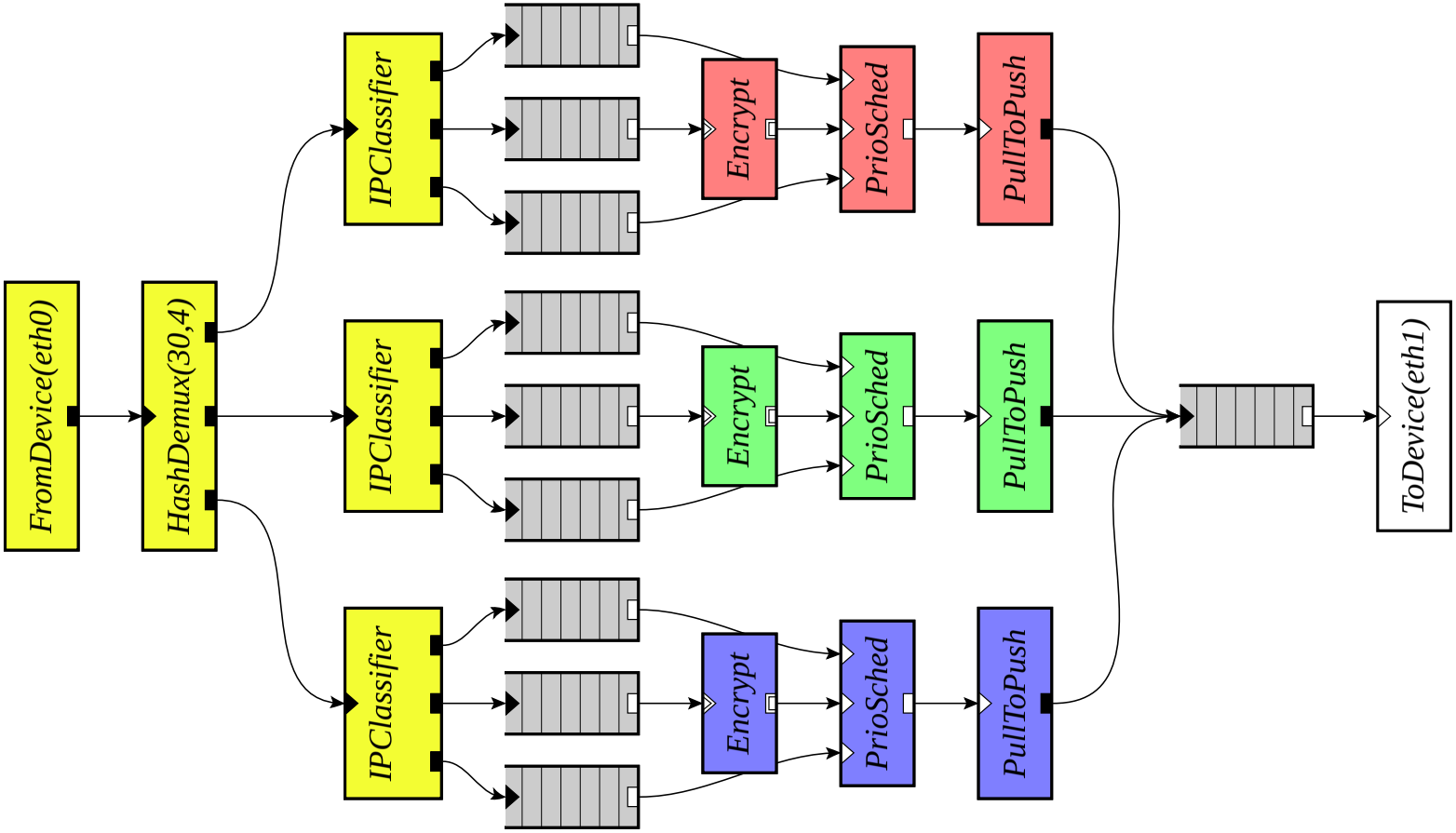
  e.g. IPsec

- Untuned configuration may not exhibit enough parallelism

Lack of Parallelism in Configuration

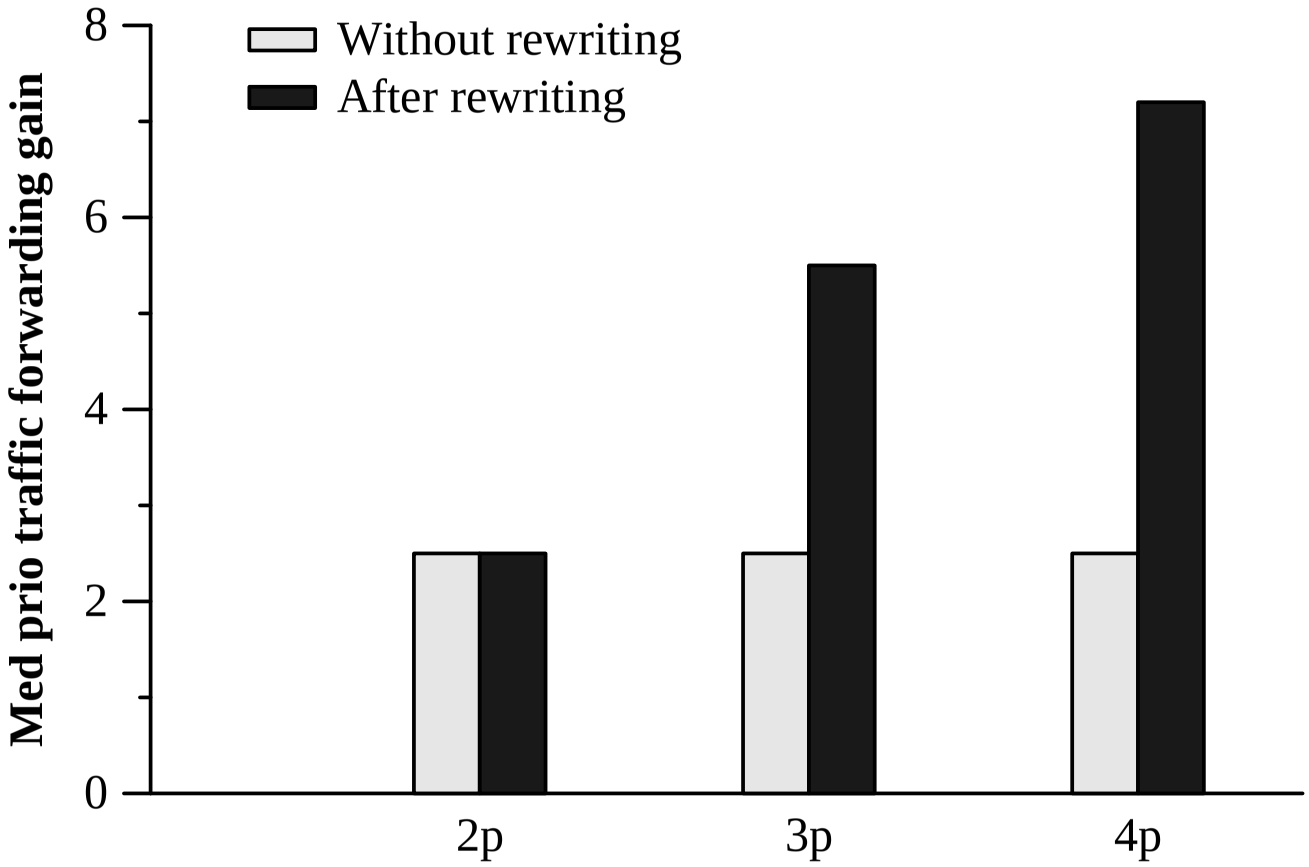


- Configuration exhibits enough parallelism for 2 CPUs
- How can we expose more parallelism ?



- `HashDemux` breaks traffic into 3 streams on per flow basis

Effects of Rewriting the Configuration



Conclusion

- A Click router configuration often has natural parallelism
- SMP Click extracts available parallelism from configuration
  - Speed up improves with higher cost of computation
  - Cost of cache misses limits speed up
- Simple changes to configuration can expose more parallelism
- Users need not to worry about synchronization issues
- SMP Click is freely available at

  http://www.pdos.lcs.mit.edu/click/