



CLICK

A NEW PROGRAMMING LANGUAGE FOR DEVELOPING SOFTWARE ROUTERS

Under the Guidance of
Prof. D Manjunath and Prof. Anirudha Sahoo



Acknowledgment

2

The graphs and figures are taken from Eddie Kohler's PhD thesis.

*Network Diagram is created on
“<http://www.lovelycharts.com/>”*



Contents

3

- Introduction
- Architecture
- The Programming Language
- Examples
- Evaluation
- Limitations
- Conclusion
- References



Introduction

4

- History - “Routers”
 - ▣ Packet forwarders => firewall, load balancer
 - ▣ Closed, Static, Inflexible Design.

- Solution – “**Click**”
 - ▣ Simplicity
 - ▣ No. of lines code
 - ▣ Solution of problems simple then normal



Architecture

5

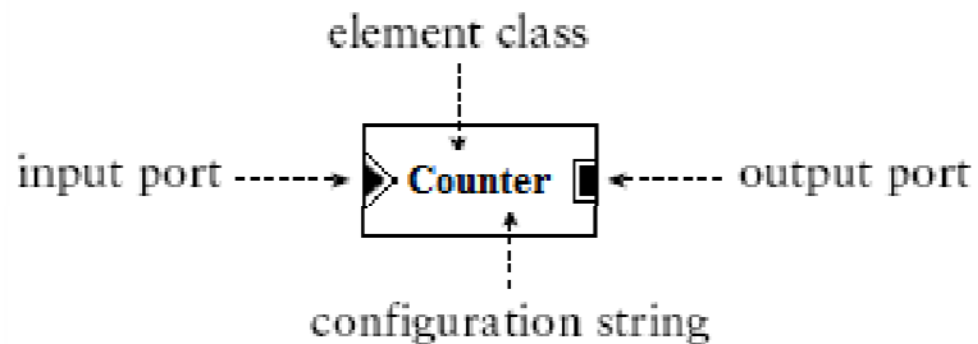
- Modular design
- Directed graph

- Element Class

- Ports

- Push
- Pull
- Agnostic

- Handlers





Click – The Programming Language

6

□ Declarative Language

□ Written in C++

□ Syntax

```
h :: Hub
q :: Queue
FromDevice(eth0) -> [0]h;
FromDevice(eth1) -> [1]h;
h[0] -> q;
h[1] -> q;
q -> toDevice(eth0);
q -> toDevice(eth1);
```

□ Examples

```
----- C++ -----
class Hub : public Element { public:
    Hub();
    ~Hub();

    const char *class_name() const
    { return "Hub"; }

    const char *port_count() const
    { return "-/="; }

    const char *processing() const
    { return PUSH; }

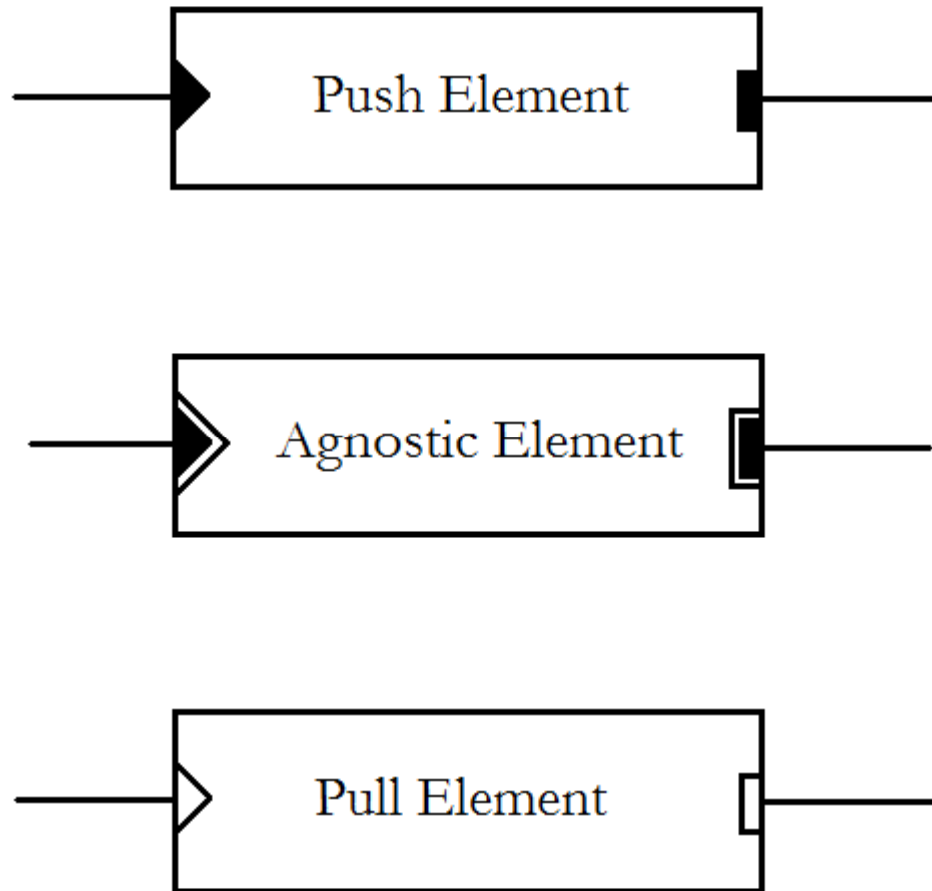
    const char *flow_code() const
    { return "#/[^#]"; }

    void push(int port, Packet* p);
};
```



Symbols and Semantics

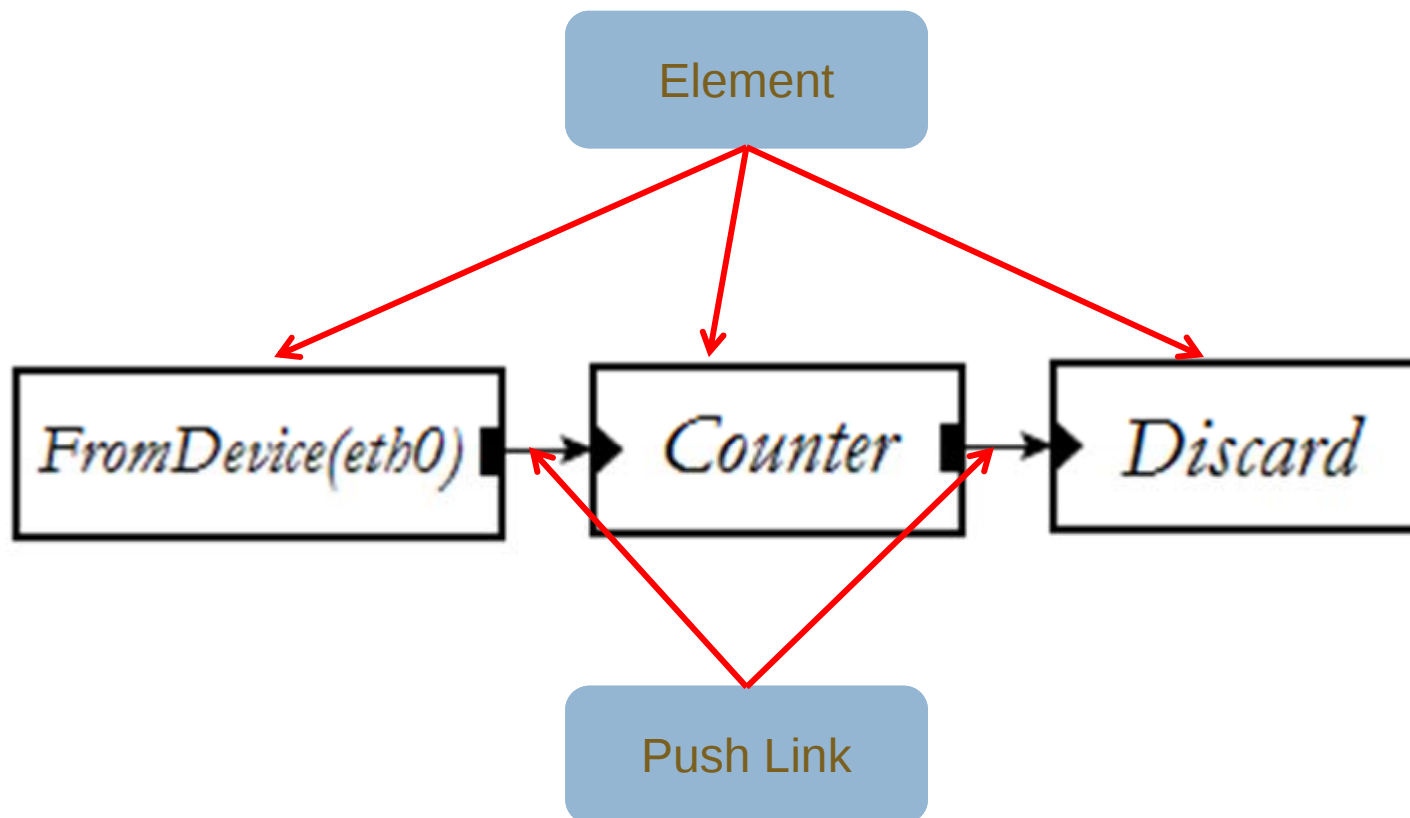
7





Example – 1

8



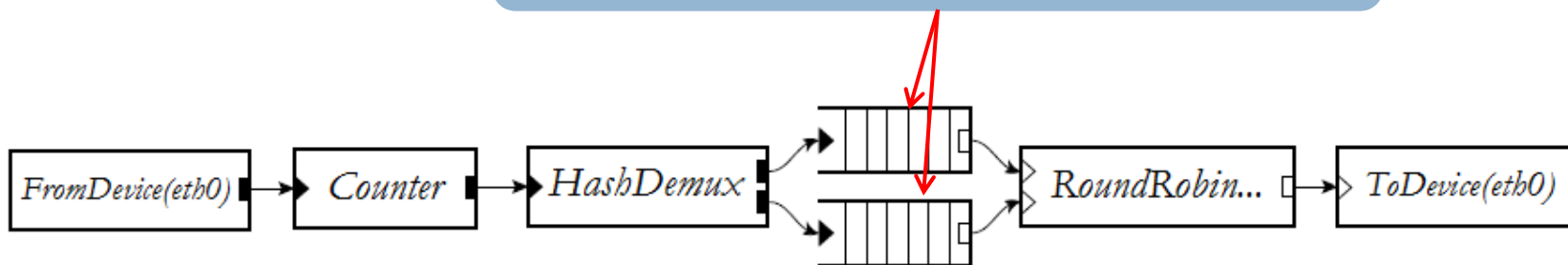
Simple Router



Example – 2

9

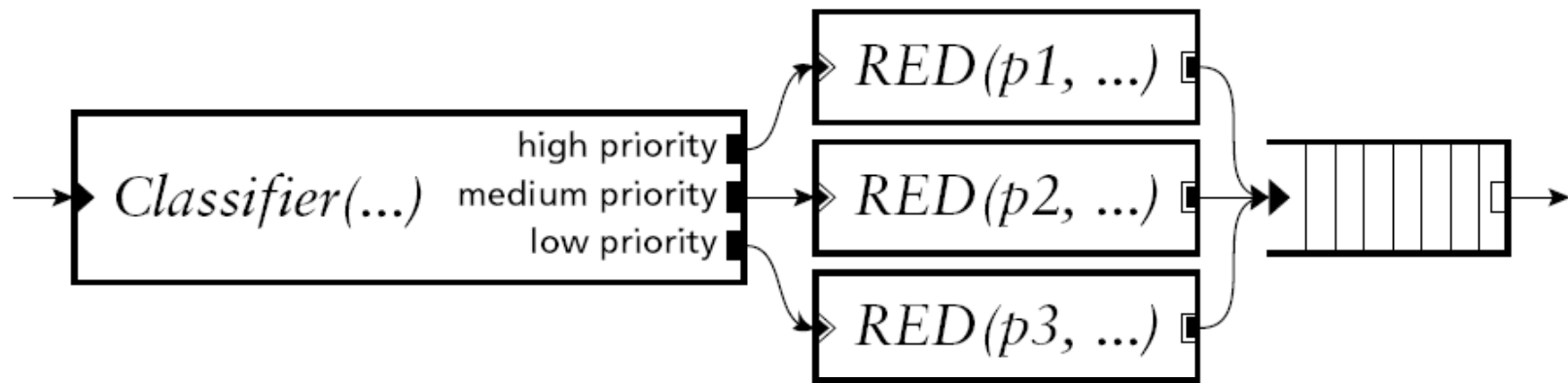
Queue is the element having both (PUSH and PULL) types of ports.



Implementing Stochastic Fairness Queuing

Example – 3

10

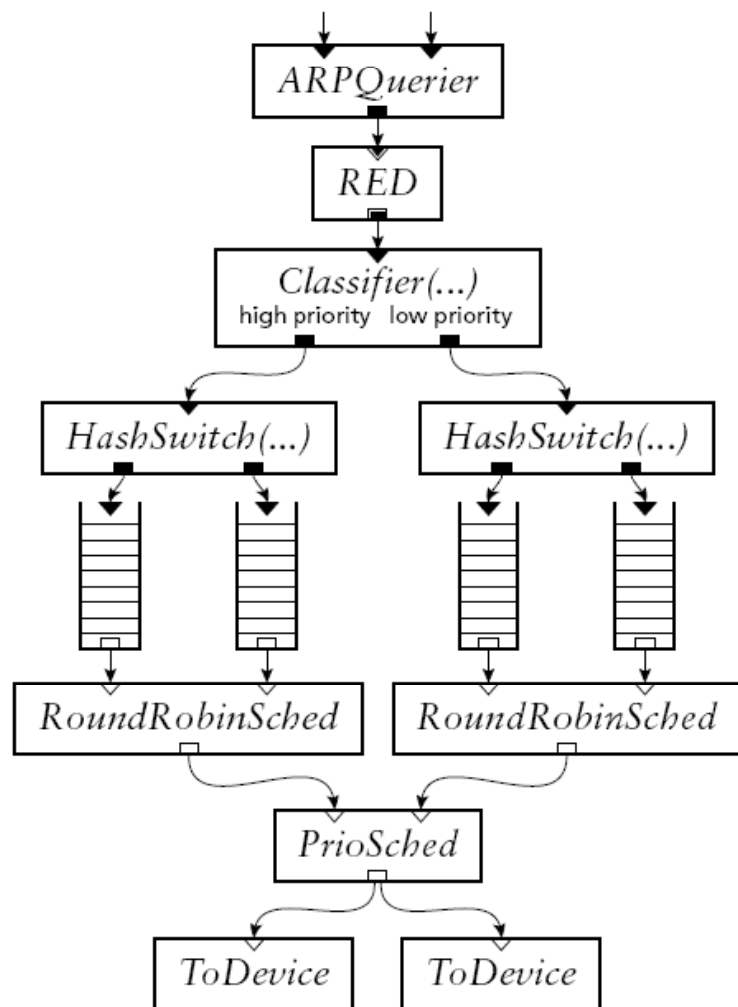


Implementing RED



Example – 4

11

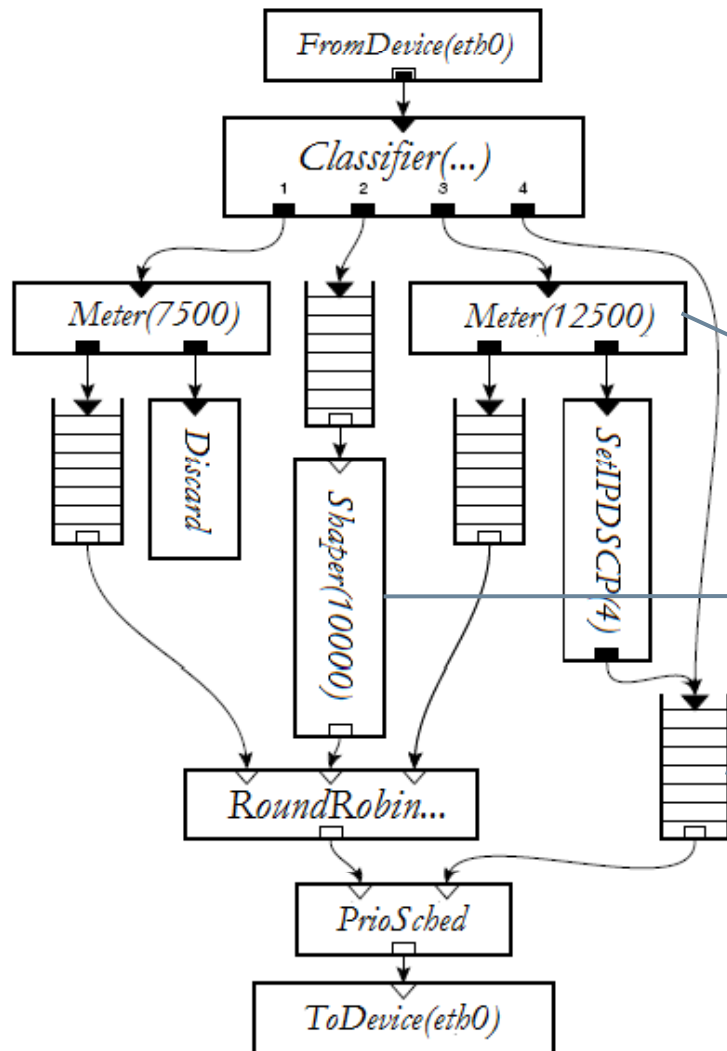


Implementing
Priority flow on the
Stochastic Fairness
Queuing with
dropping mechanism



Example – 5

12



Differentiated Services Algorithm

Will send 12,500 packets/second on its first output port other on second output port.

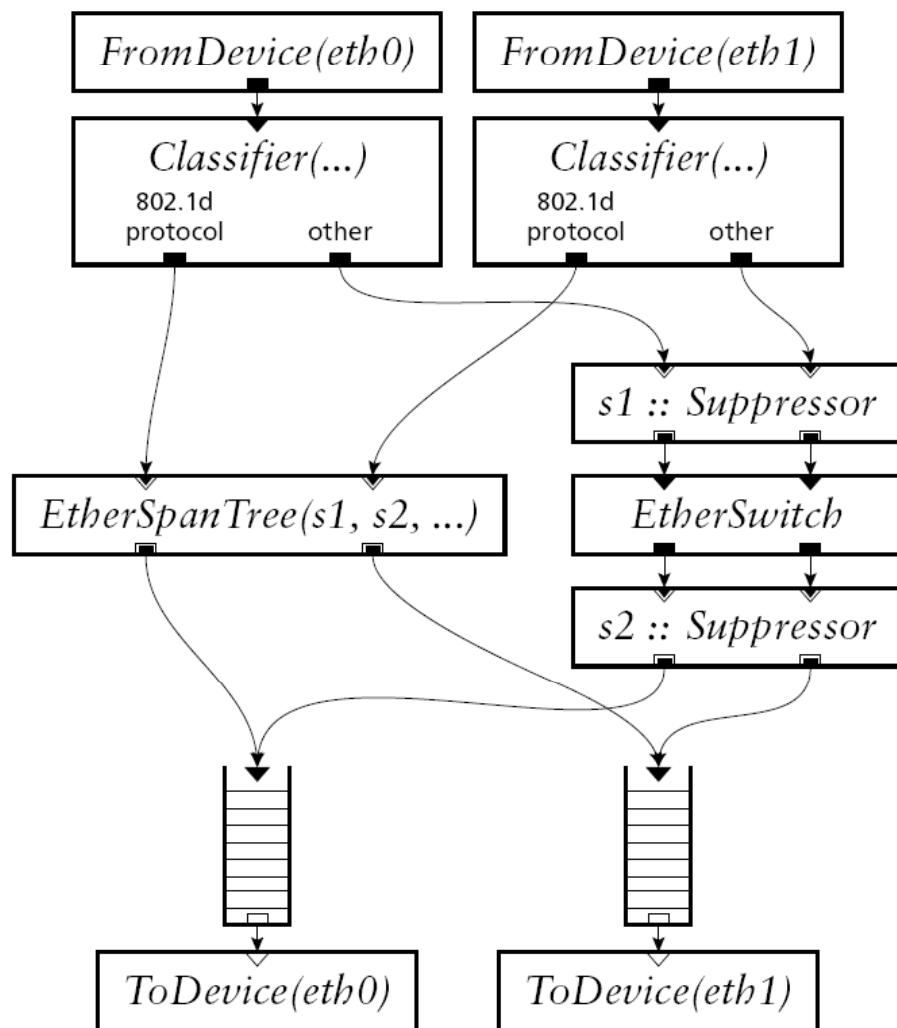
Will pull 10,000 packets/second queue will determine which packets to drop, if busy flow.

Best effort flow queue.



Example – 6

13

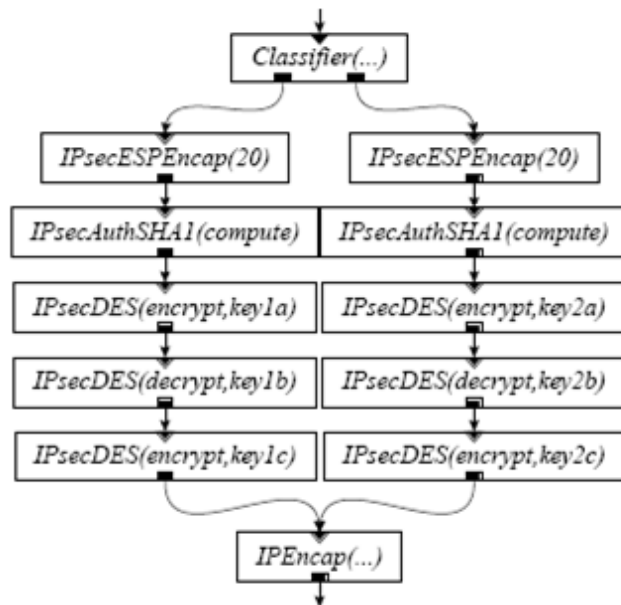


Ethernet Switch

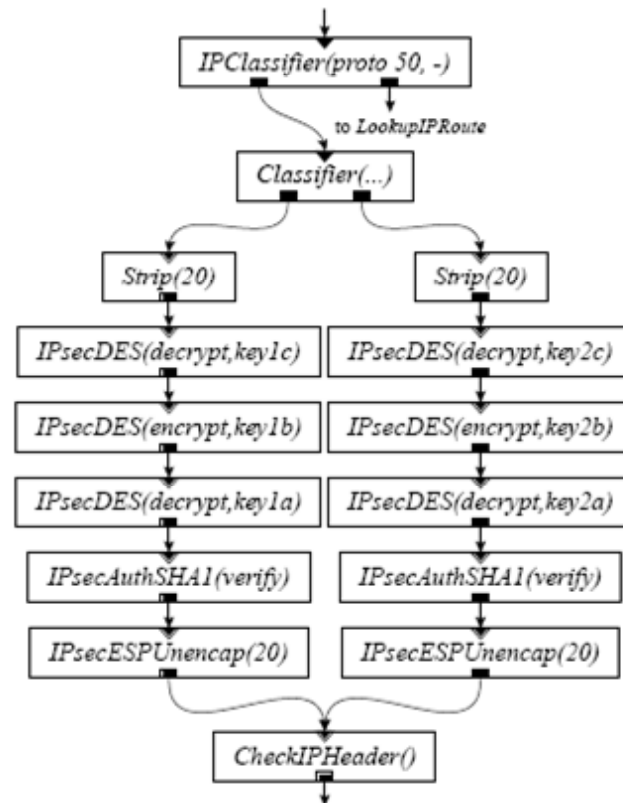
This works as a 802.1d standard learning bridge between two networks connected to eth0 and eth1 ports.

Example – 7

14



IPSec VPN Tunnel Encryption configuration.



IPSec VPN Tunnel Decryption configuration.

IPSec Implementation

Figure Source [4]

15

```

graph TD
    subgraph Path0 [Path for eth0]
        FD0[FromDevice(eth0)] --> C0[Classifier(...)]
        C0 -- ARP queries --> ARPR0[ARPResponder 1.0.0.1 ...]
        ARPR0 --> Q0[Queue]
        C0 -- ARP responses --> AQ0[ARPQuerier]
        AQ0 --> TD0[ToDevice(eth0)]
        C0 -- IP --> P1[Paint1]
    end

    subgraph Path1 [Path for eth1]
        FD1[FromDevice(eth1)] --> C1[Classifier(...)]
        C1 -- ARP queries --> ARPR1[ARPResponder 2.0.0.1 ...]
        ARPR1 --> Q1[Queue]
        C1 -- ARP responses --> AQ1[ARPQuerier]
        AQ1 --> TD1[ToDevice(eth1)]
        C1 -- IP --> P2[Paint2]
    end

    P1 --> S[Strip14]
    P2 --> S
    S --> CHI[CheckIPHeader]
    CHI --> GIPA[GetIPAddress16]
    GIPA --> LIPR[LookupIPRoute]

    LIPR --> L[Linux]
    LIPR --> DB0[DropBroadcasts]
    LIPR --> DB1[DropBroadcasts]

    DB0 --> PT1[PaintTee1]
    PT1 --> IOW0[IPGWOptions 1.0.0.1]
    IOW0 --> FIS0[FixIPSrc 1.0.0.1]
    FIS0 --> DPT0[DecIPTTL]
    DPT0 --> IF0[IPFragmenter 1500]
    IF0 --> AQ0[ARPQuerier 1.0.0.1 ...]
    AQ0 --> TD0

    DB1 --> PT2[PaintTee2]
    PT2 --> IOW1[IPGWOptions 2.0.0.1]
    IOW1 --> FIS1[FixIPSrc 2.0.0.1]
    FIS1 --> DPT1[DecIPTTL]
    DPT1 --> IF1[IPFragmenter 1500]
    IF1 --> AQ1[ARPQuerier 2.0.0.1 ...]
    AQ1 --> TD1

    IF0 --> IE0[ICMPError]
    IF1 --> IE1[ICMPError]
    DPT0 --> IE2[ICMPError]
    DPT1 --> IE3[ICMPError]
    IOW0 --> IE4[ICMPError]
    IOW1 --> IE5[ICMPError]
    PT1 --> IE6[ICMPError]
    PT2 --> IE7[ICMPError]
    
```



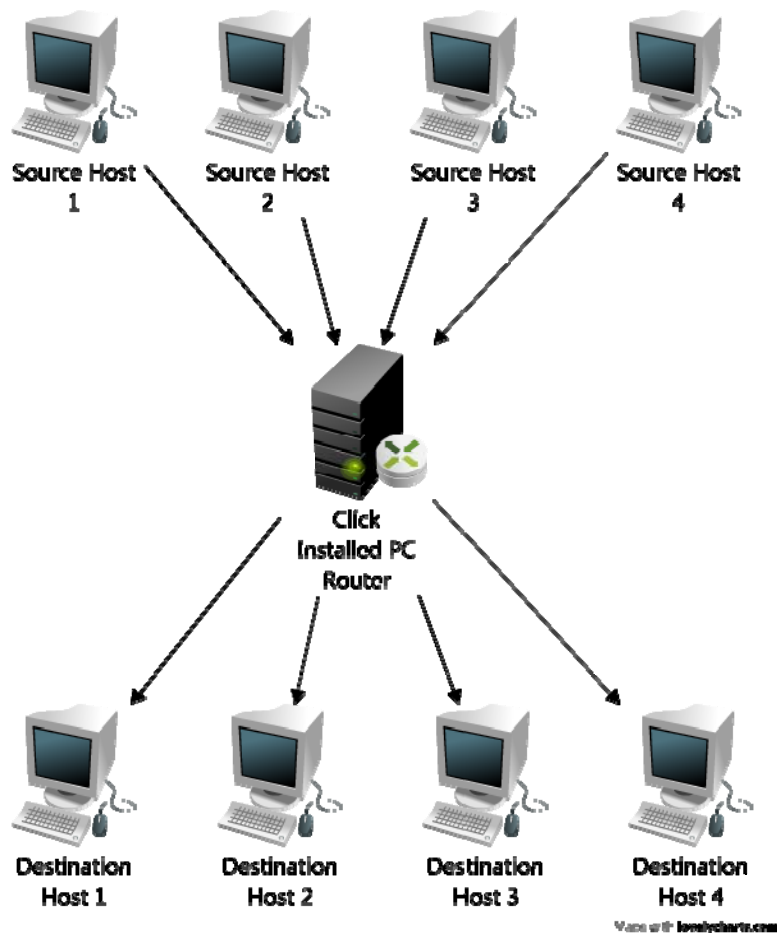
Evaluation

16

- Forwarding path
- Performance of the “**Click**” diffserv configuration
- Comparison of IP router and non-IP router created with click
- CPU time breakdown

Experimental Setup

17

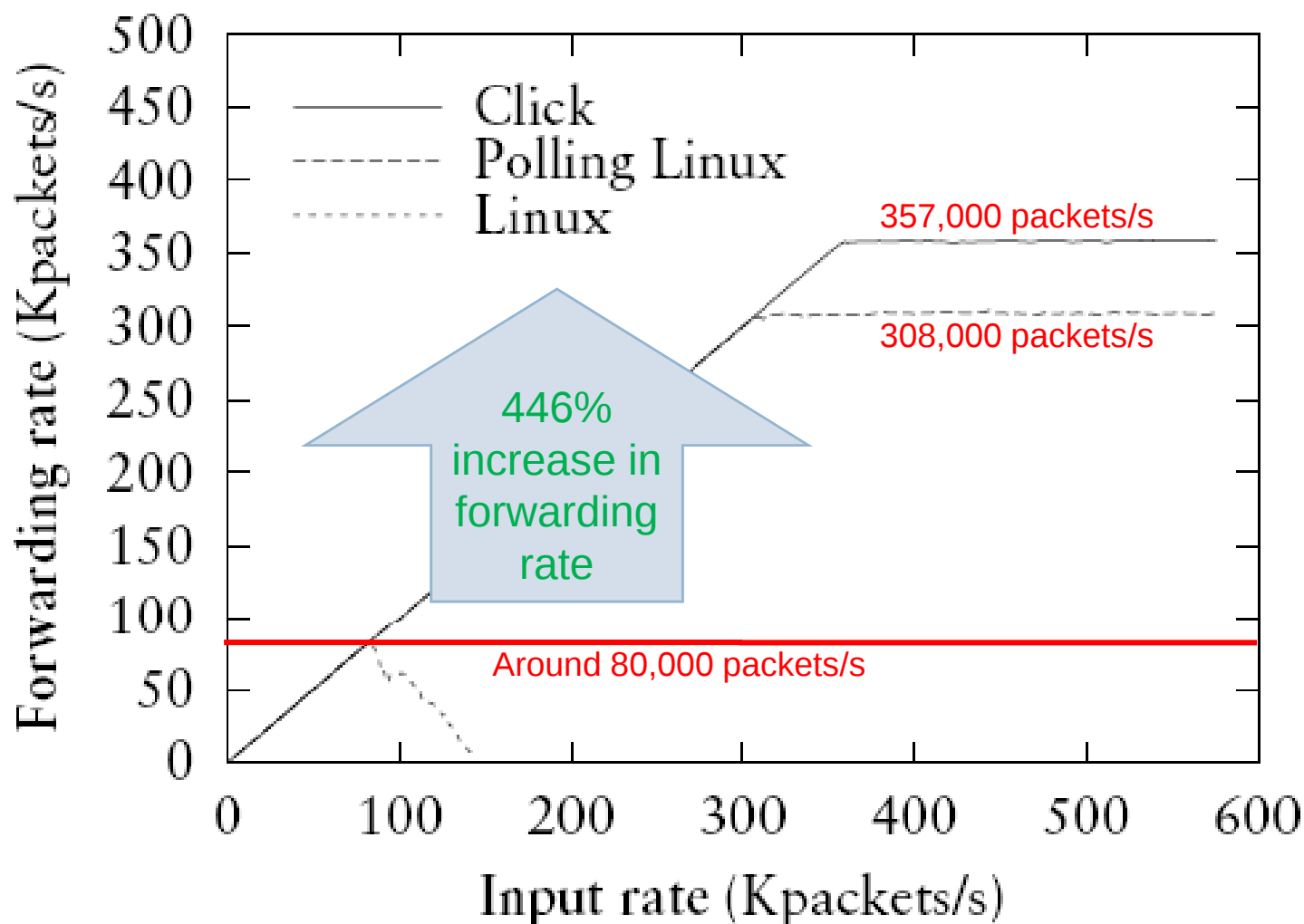


- All are having DEC 21140 Tulip 100 Mbit/s PCI Ethernet Controllers.
- Router Host
 - 700 MHz Intel Pentium III CPU
 - Eight Ethernet Controllers are on multi-port cards split across the motherboard's two independent PCI buses.
- Source Hosts
 - 733 MHz Pentium III CPUs
 - Generates the 64-byte packets per second of UDP flow.
- Destination Hosts
 - 200 MHz Pentium Pro CPUs
 - Counts and Discards the packets.



Forwarding Path

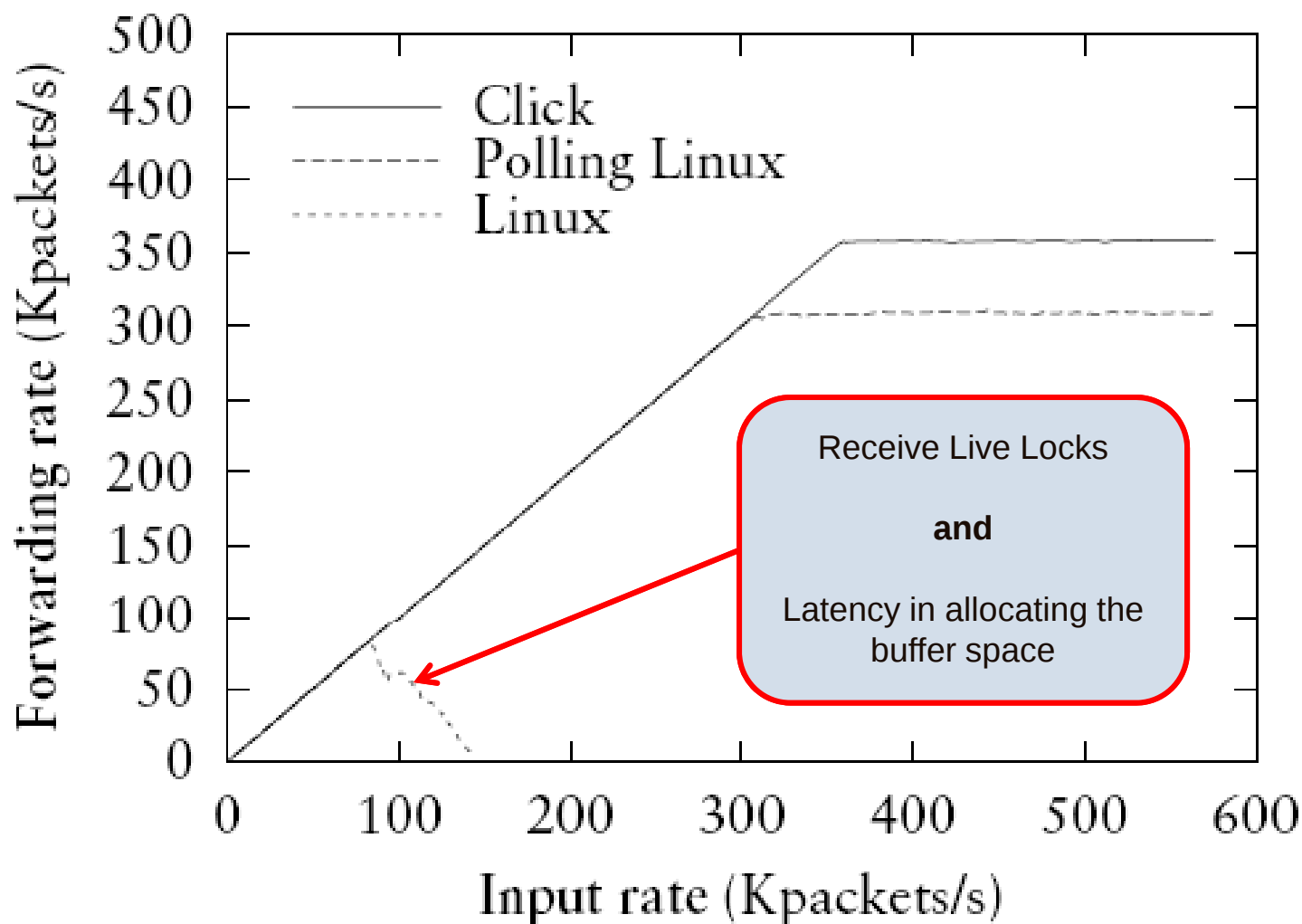
18





Forwarding Path

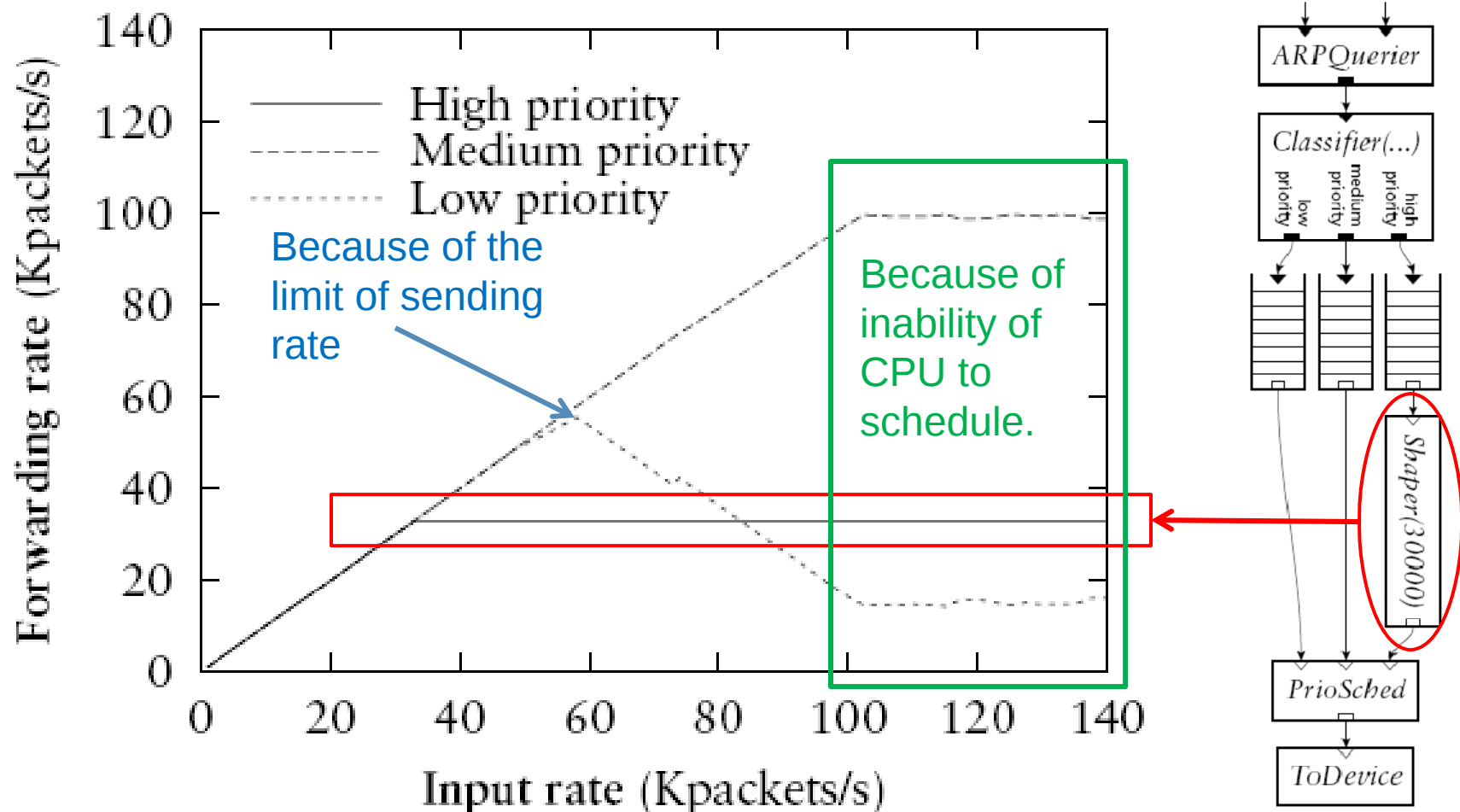
19





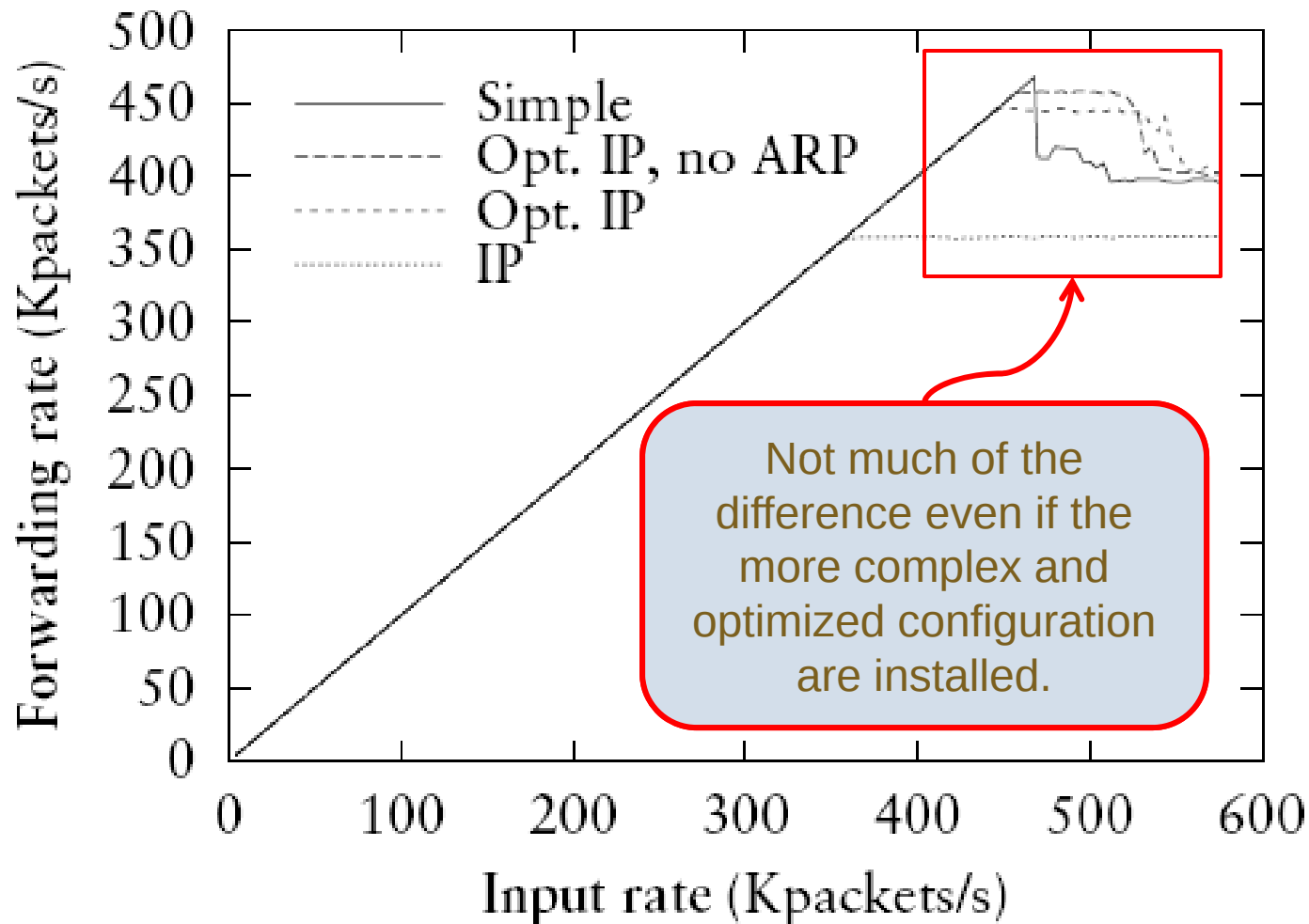
Performance of the Click diffserv configuration

20



Comparison of IP router and non-IP router created with click

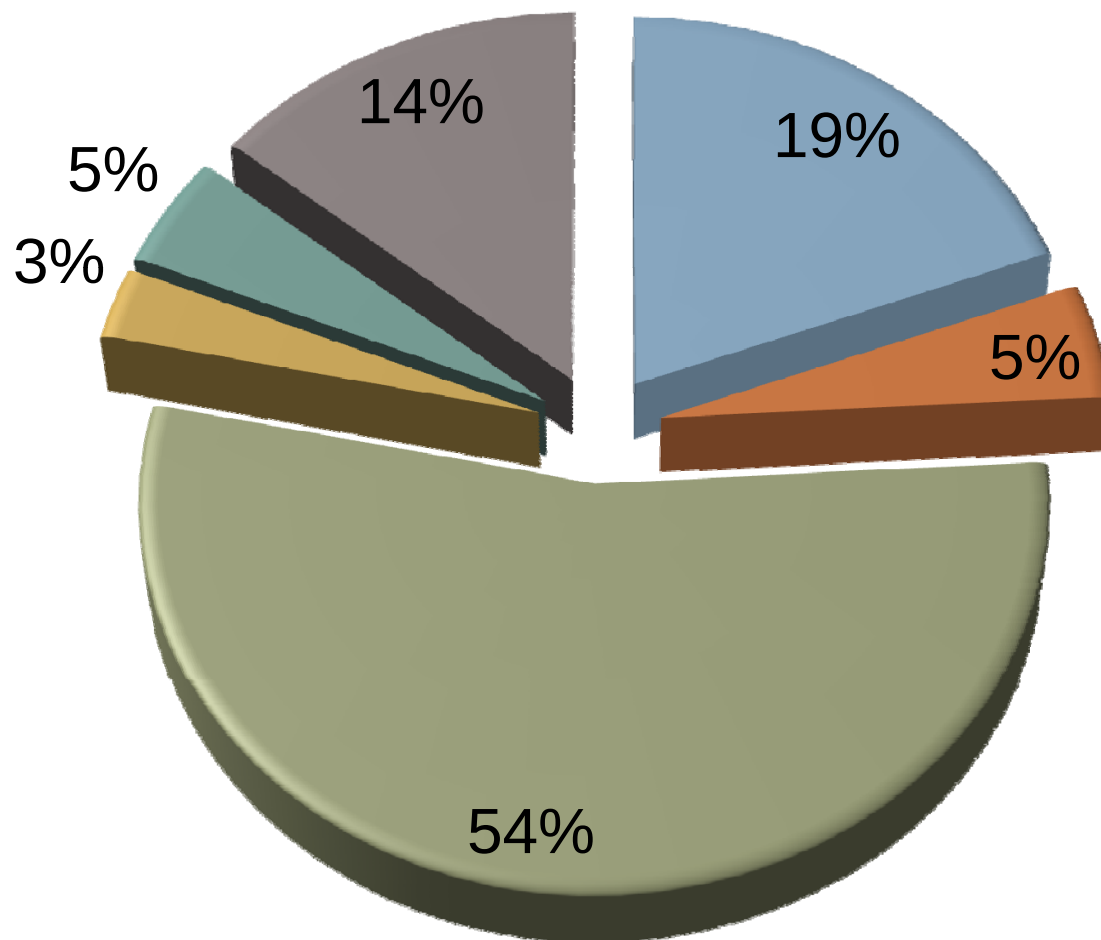
21





CPU Time Breakdown

22



Total Time : 2905 ns

Push Time : 1572 ns

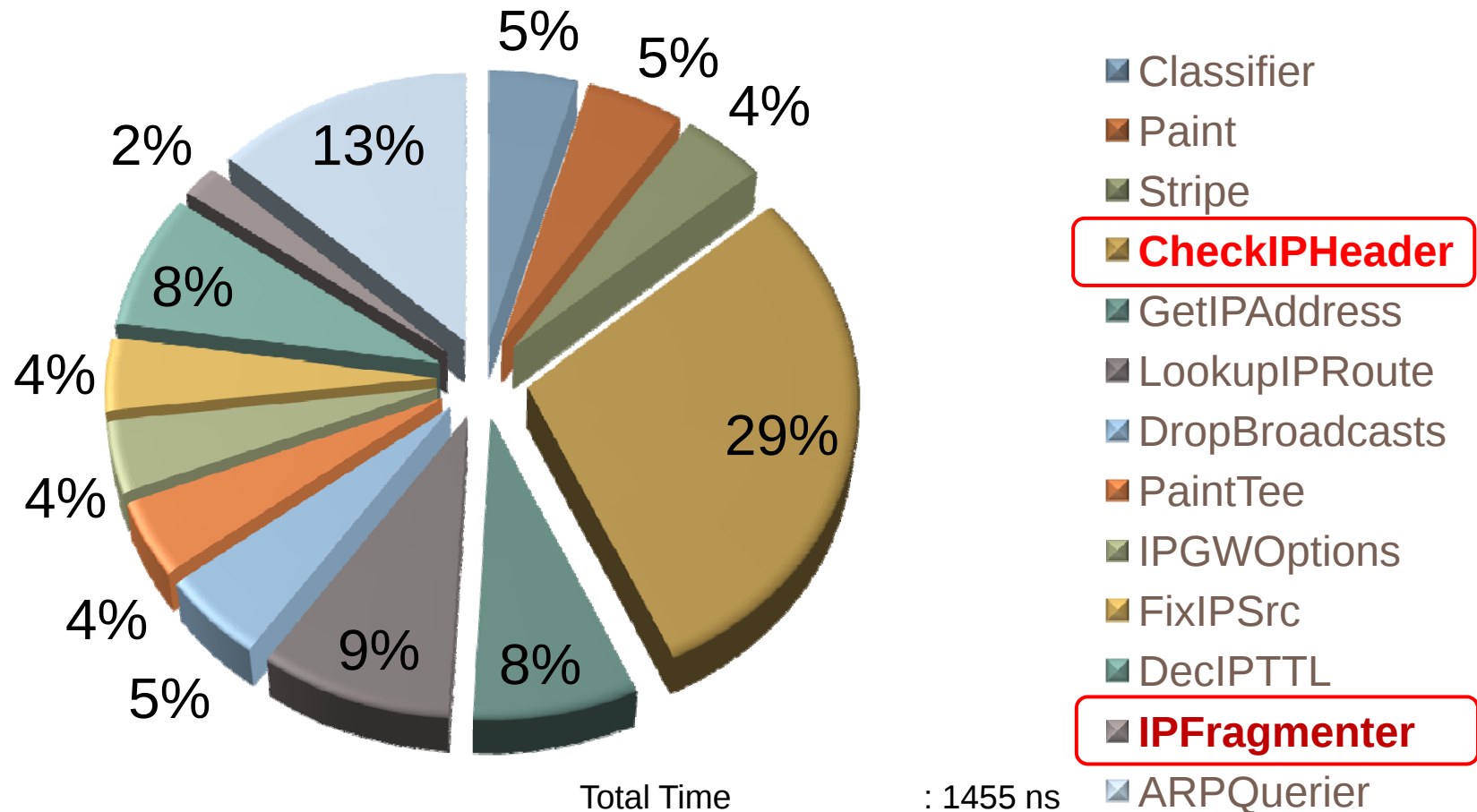
- Polling packet
- Refill receive DMA ring
- **Click forwarding path: push**
- Click forwarding path: pull
- Enqueue packet for transmit
- Clean transmit DMA ring

CPU Time Breakdown (Cont...)



23

Per element execution time



Total Time : 1455 ns
CheckIPHeader Time : 457 ns



Limitations

24

- Limitations of the tool
 - ▣ “**Click**” cannot be used to create the coarse-grained elements.
 - Coarse-grained elements are required when control or data flow doesn't match the flow of packets.
 - Ex. BGP.
 - ▣ “**Click**” cannot schedule the CPU per individual flow.
 - It is using stride algorithm for CPU scheduling.
- Limitations of the language
 - ▣ Compound element classes
 - Ex. The customized handlers implementation is not possible.
 - ▣ Compound elements are an imperfect abstraction mechanism.
 - They do not hide themselves from the user.
 - They are strictly less powerful than native element classes.



Conclusion

25

- Simple design
- Readable language configurations
- Better performance in smaller networks
- Easy to find faults because of modularity



References

26

- [1] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(4), November 2000.
- [2] The Click Modular Router Project: <http://read.cs.ucla.edu/click/>
- [3] Eddie Kohler, Click for Measurement, *UCLA Computer Science Department Technical Report TR060010*, February 2006
- [4] Benjie Chen and Robert Morris, Flexible Control of Parallelism in a Multiprocessor PC Router, *USENIX 2001 Annual Technical Conference*, June 2001.