COST 605 ,
September 24-26, 2008, Zürich, Switzerland

DiCAP - An Architecture for Distributed Packet Capturing

Cristian Morariu

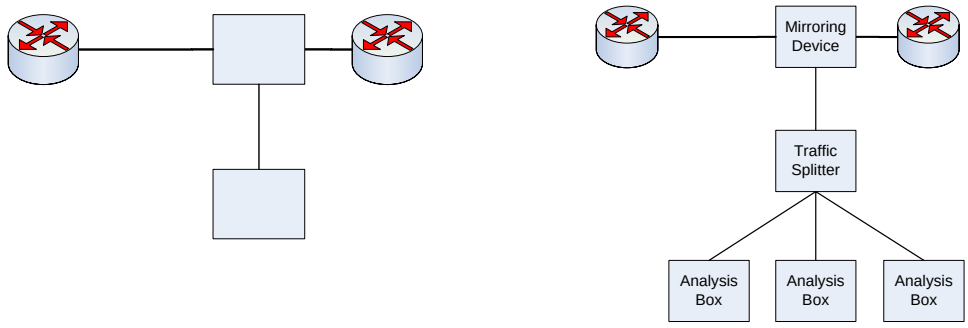
Department of Informatics IFI, Communication Systems Group CSG, University of Zürich

Motivation
Design and Implementation
Evaluation
Concluding Remarks

Overview

- Most network monitoring tasks require packet inspection
- Network monitoring
  - *Live*
    - packets are inspected in real-time
    - data is dropped after inspection
    - *e.g.* traffic accounting, IDS
- High packet rates reduce the time available to handle a single packet
- Solutions for high packet rates:
  - packet sampling
    - decreased measurement accuracy
  - dedicated hardware
    - Expensive

Traditional Architecture for Traffic Analysis



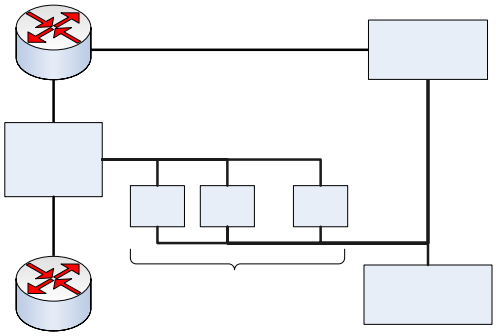
- Simple
- Easy to deploy
- Not scalable due to a single analysis box

- More complex
- More scalable with respect to performance
- The single point of failure still present

Motivation

- Build a distributed architecture for IP traffic capturing
  - Avoid a single point of failure
  - Avoid dedicated hardware
  - Based on stock, inexpensive PCs
  - Allows the increase of packets that can be captured

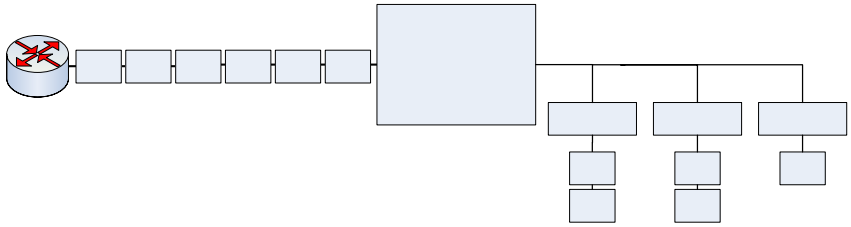
DiCAP Architecture



- Each node receives all packets
- Node coordinator decides the capture policy on each capture node
- The capture nodes selects which packets to capture packets so that no packet is captured by two nodes
- Packet data is sent to the packet data analyzer for further analysis

Selection Policies (1)

- Round robin selection
  - Node coordinator introduces control packets in the mirrored traffic
  - Capture nodes are logically organized in a chain
  - After each packet, capture responsibility shifts to the next node in the chain
  - Node coordinator configures the chain



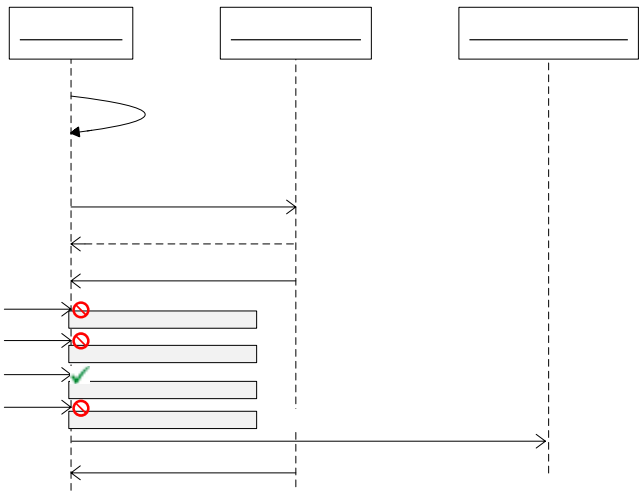
C1

Capt

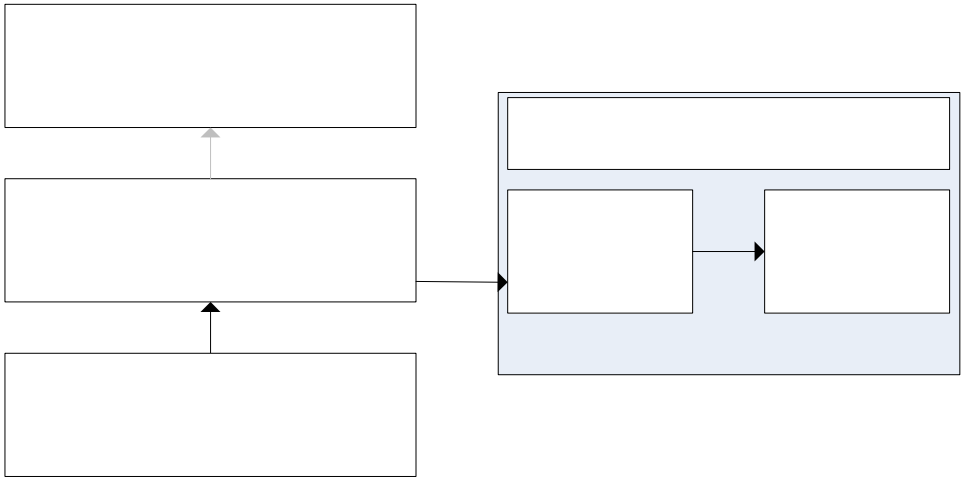
Selection Policies (2)

- Hash-based selection
  - A hash function is applied on packet headers
    - Current implementation uses IP identification field
  - Each node is responsible with a particular range of hash values
  - Each packet is captured by the node responsible with the respective range of hash values
- Advantages:
  - Easier synchronization
  - No need of control packet injection
- Disadvantages:
  - More computation needed for hash calculation

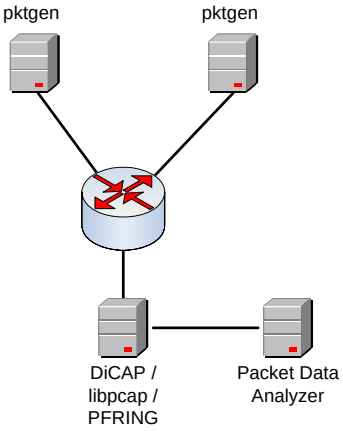
Capture Example in Round-Robin Selection



DiCAP Implementation



Evaluation Testbed



- Two nodes used for traffic injection using Linux pktgen
- One DiCAP node for testing capture performance of a single node
  - 1G Ethernet Broadcom NetXtreme BCM5721 card
- Packet data is sent to the packet data analyzer for further analysis
- One packet data analyzer

Linux

NIC D

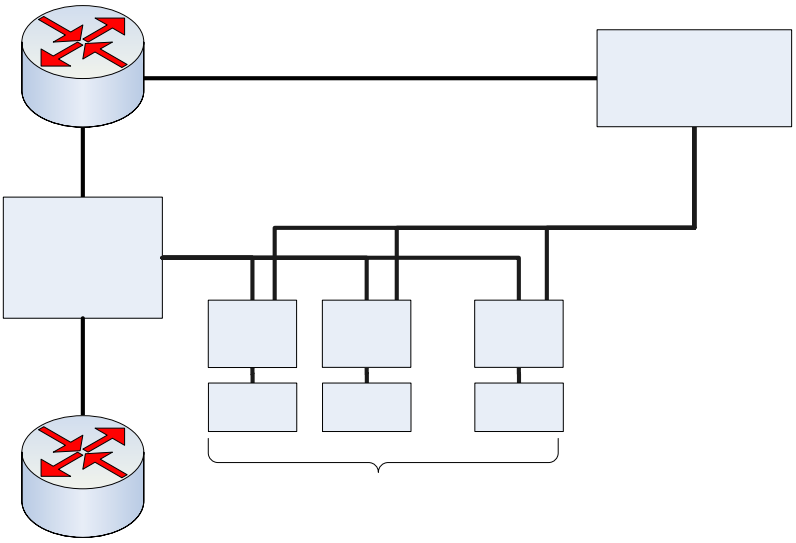
DiCAP Capture Evaluation

Packet Rate	libpcap Loss	PFRING Loss	DiCAP Loss
119 Kpps	0%	0%	0%
232 Kpps	10%	1%	0%
380 Kpps	75%	28%	0%
492 Kpps	90%	83%	0%
620 Kpps	93%	96%	0%

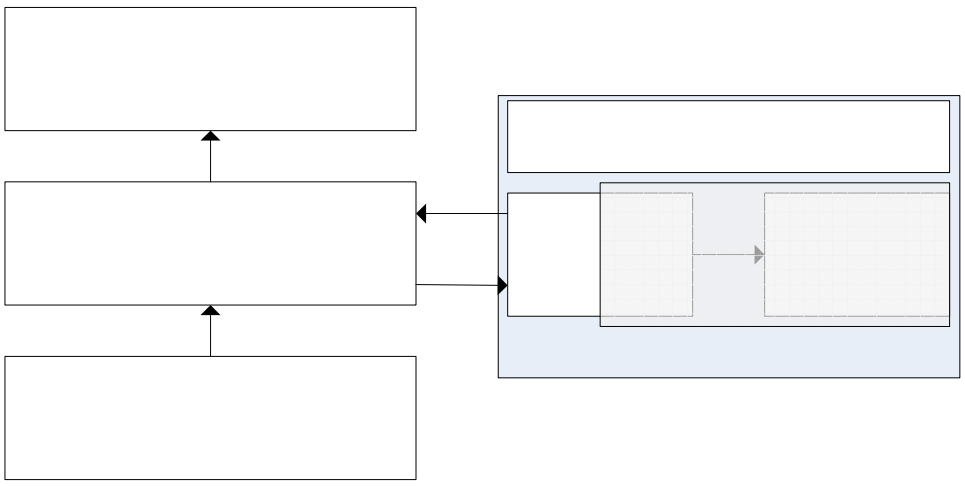
Operation Modes

- Capture Mode
  - Packet headers are captured and forwarded to a packet analyzer
  - No possibility to analyze payload of packets
- Distribution Mode
  - DiCAP does not capture anything
  - Decides which packets are forwarded to the higher levels
  - Capture takes place in user space applications
  - Allows parallel use of *libpcap*

DiCAP in Distribution Mode

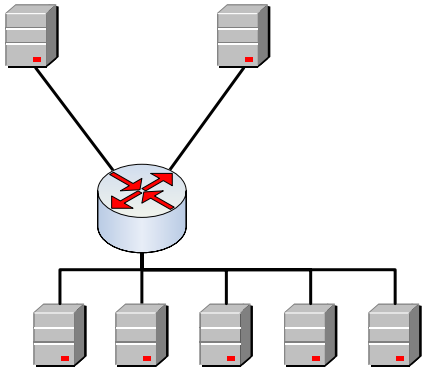


DiCAP Implementation – Distribution Mode



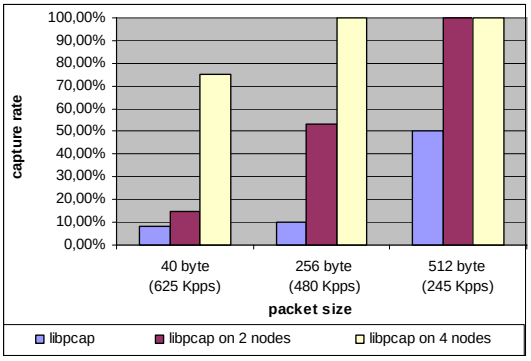
Mirrorin
Device

Evaluation Testbed



- Two nodes used for traffic injection using Linux pktgen
- One libpcap node for testing
- Up to 4 DiCAP nodes in distribution mode

Distribution Mode Evaluation



- 3 different tests were performed

1. traditional libpcap on a single PC
2. DiCAP in distribution mode on 2 PCs
3. DiCAP in distribution mode on 4 PCs

- DiCAP can improve the capture performance up to 700% when 4 PCs are used in parallel

Concluding Remarks

- DiCAP can be used to distributedly capture packets on a high-speed link
- It may be used to allow a distributed deployment of libpcap-based applications running on common, inexpensive PCs
- It may significantly improve libpcap performance
- Very simple design, easy to implement in hardware.
- Further investigation on using other hash functions may lead to performance improvement and better load ballance

Thank You!