Seminar

Internet Measurement

Erhöhung der Betriebssystemeffizienz bei GBit-Netzwerkpaketerfassung

Donnerstag, 26. Feb. 2009

Mohannad Alnablsi

Technische Universität Berlin

Motivation zu GBit-Messungen

- Zunahme der Größe und Geschwindigkeit von Rechnernetzwerken
- GBit-Netzwerke sind Standard

- erhöhter Netzwerksverkehr bzw. erhöhte Paketanzahl
  ⇒Anpassung der Messinstrumente erforderlich

- Hardwareansätze:
  - Optimierte Netzwerkkarten
- Softwareansätze:
  - Optimierte Betriebssysteme

Überblick

- Paketerfassung bei Betriebssystemen
- Problem diskutieren
  - Kernel und Benutzerraum
  - Interrupt-Live-Lock und Device-Polling
- PF_RING
- nCap
  - nCap vs. PF_RING
- Zusammenfassung

Erster Test: Datenerfassung

- Netzwerk mit 100 Mbps.
- Rechner mit ~500 MHz CPU als Paketerfasser
- durch Hardware Messung der Anzahl der tatsächlichen Pakete
- Software basiert auf [libpcap]

- Ergebnisse:

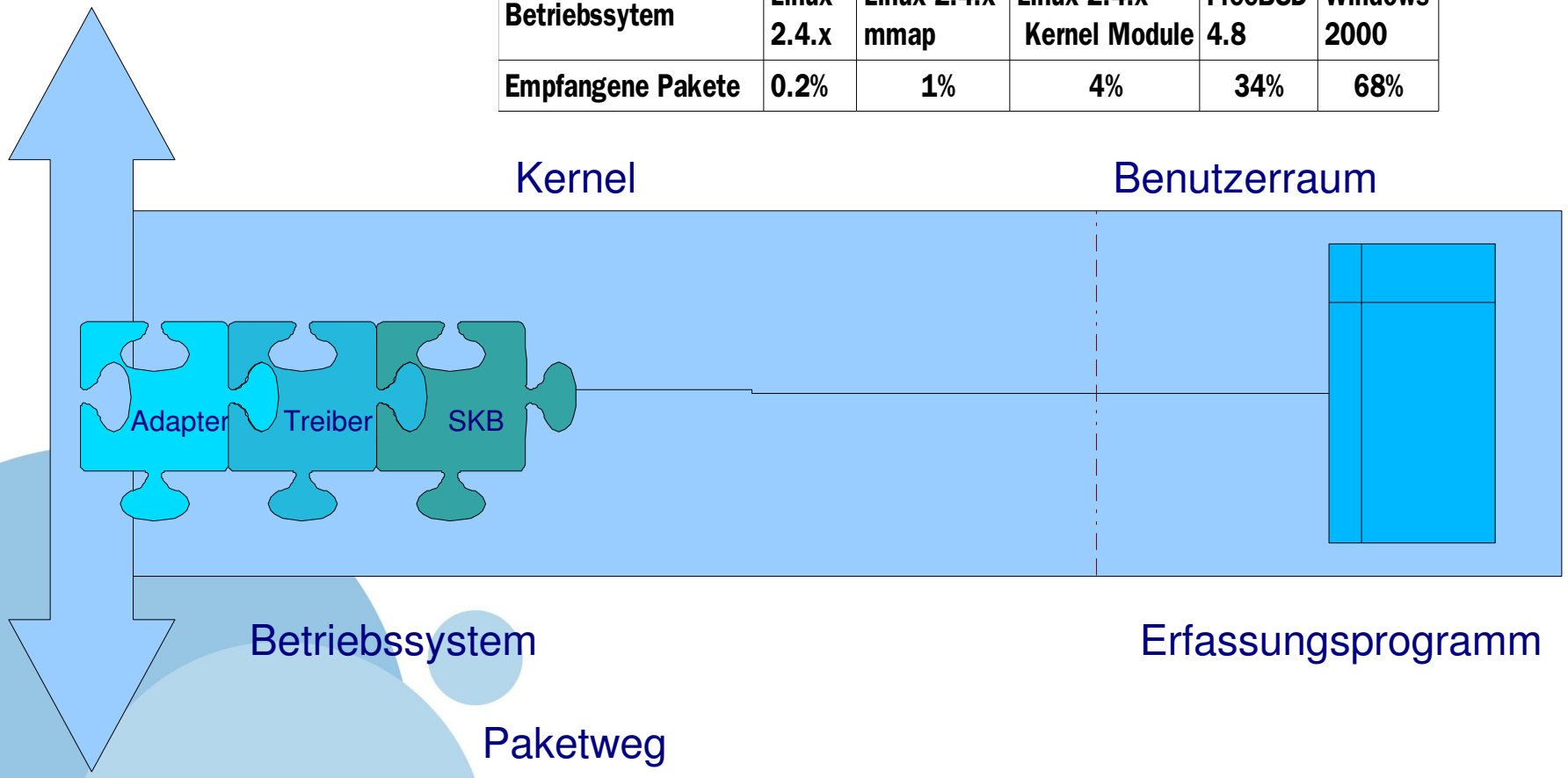
Betriebssytem	Linux 2.4.x	Linux 2.4.x mmap	Linux 2.4.x Kernel Module	FreeBSD 4.8	Windows 2000
Empfangene Pakete	0.2%	1%	4%	34%	68%

Warum erbringt Linux schlechte Leistung ?

Quelle: [Deri BDP]

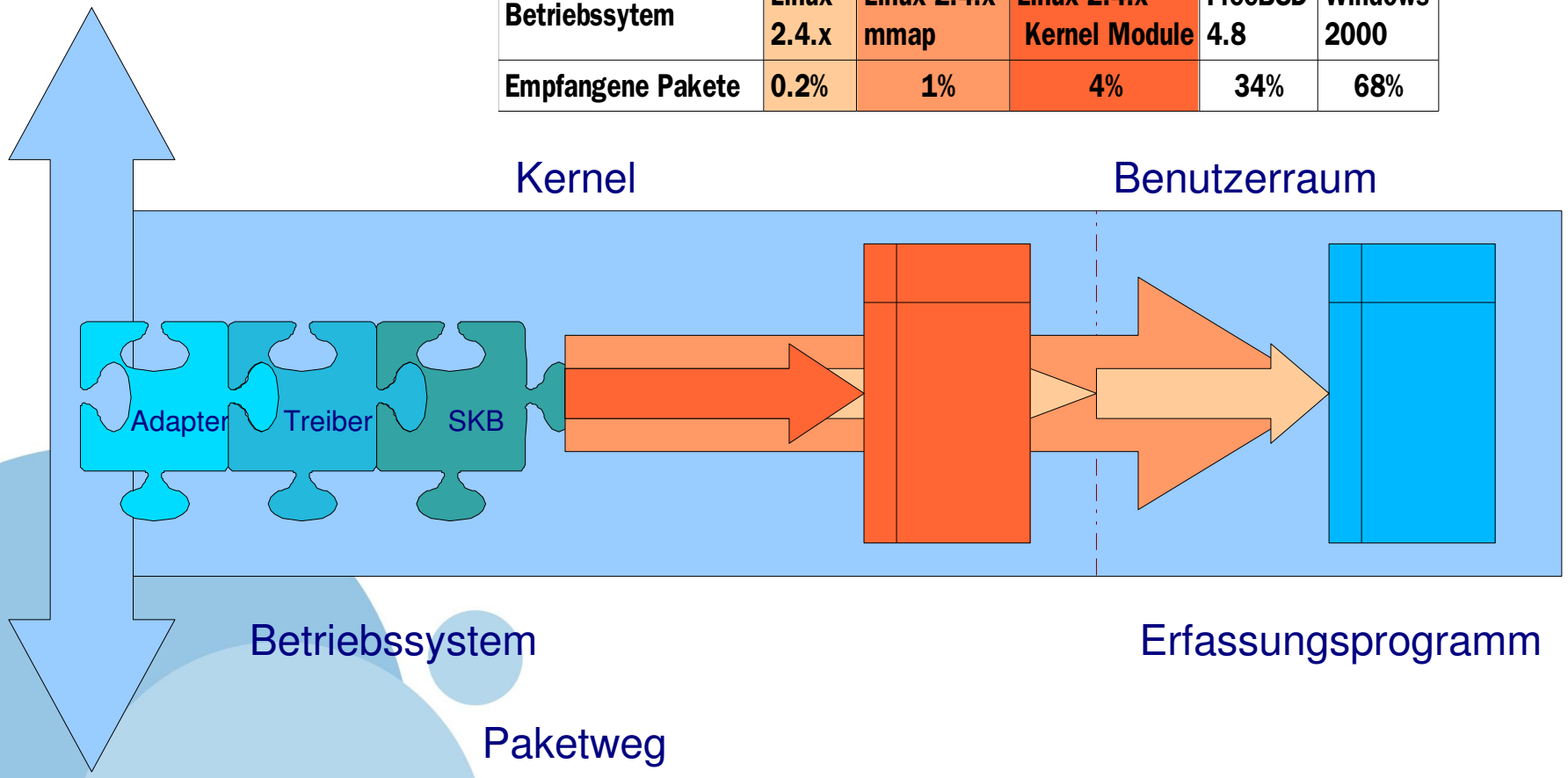
Erster Test: Datenerfassung

Betriebssytem	Linux 2.4.x	Linux 2.4.x mmap	Linux 2.4.x Kernel Module	FreeBSD 4.8	Windows 2000
Empfangene Pakete	0.2%	1%	4%	34%	68%



Erster Test: Datenerfassung

Betriebssytem	Linux 2.4.x	Linux 2.4.x mmap	Linux 2.4.x Kernel Module	FreeBSD 4.8	Windows 2000
Empfangene Pakete	0.2%	1%	4%	34%	68%



Device-Polling

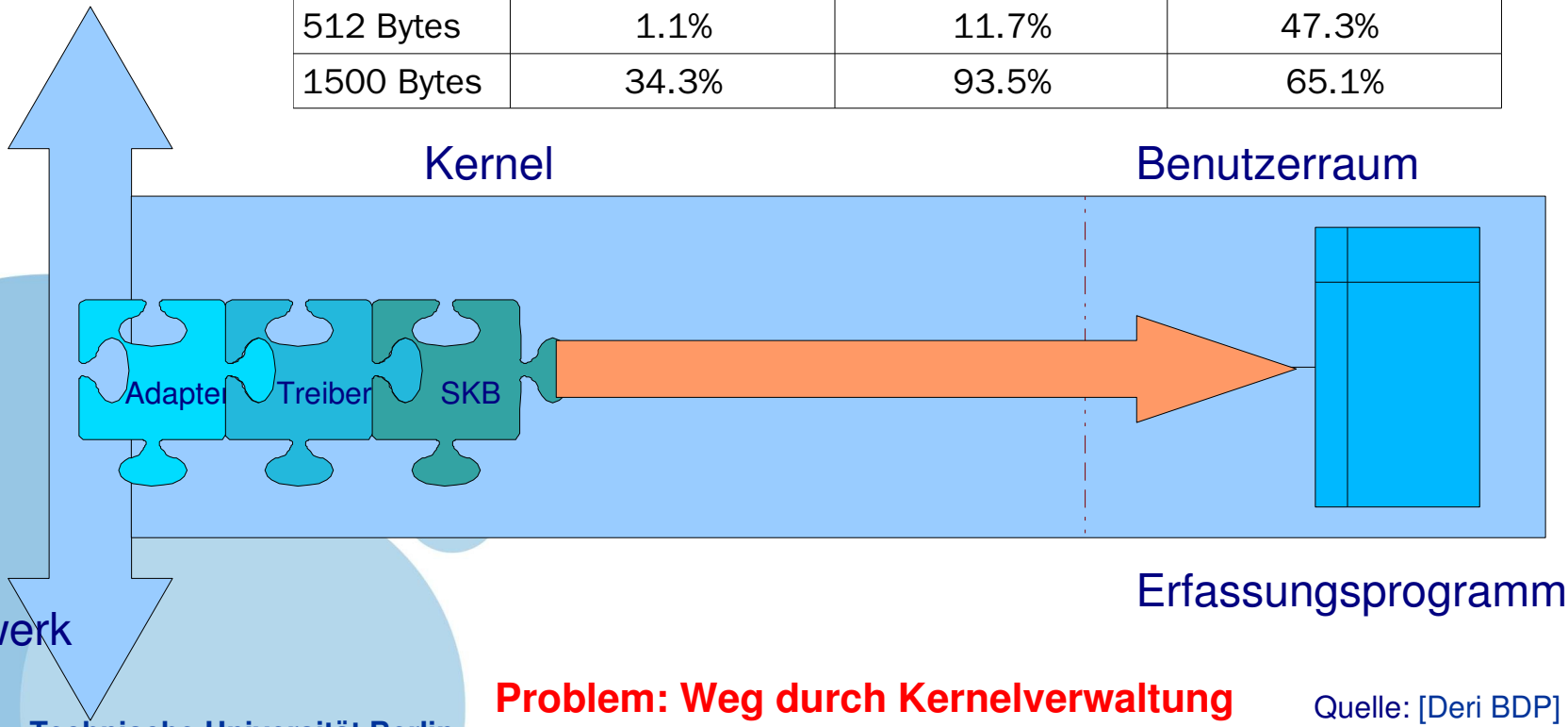
Lösung für das Interrupt-Live-Lock-Problem :
Device Polling

- Maskieren des Interrupts bei der Ankunft eines Paketes
- Treiber periodisch aus dem Netzwerkadapter lesen lassen
- am Ende der zu lesenden Pakete ⇒ Interrupt wieder aktivieren

Zweiter Test: GBit und Paketgröße

1 GBit Netzwerkkonfiguration

Paketgröße	Linux NAPI Standard libpcap	Linux NAPI mmap libpcap	FreeBSD Polling
64 Bytes	2.5%	14.9%	97.3%
512 Bytes	1.1%	11.7%	47.3%
1500 Bytes	34.3%	93.5%	65.1%



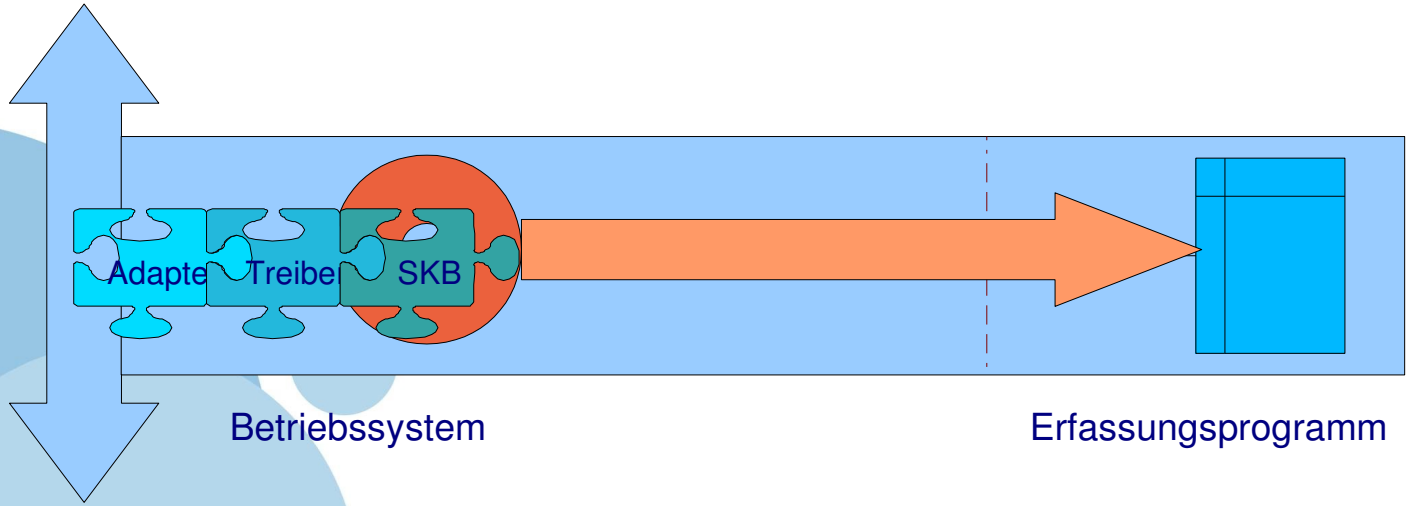
Problem: Weg durch Kernelverwaltung

Quelle: [Deri BDP]

PF_RING

Ziel: Lösung für den Kerneloverhead und
Ausnutzung von Device-Polling und mmap()

- Einführen eines Ringpuffers
- Entwurf einer neuen Socketart
- Realisierung von PF_RING als Kernelmodul



PF_RING

- kontinuierliche Speicherung von Paketen ohne unnötige Speicherreservierung
- Pakete werden nicht mittels Kernelverwaltung behandelt
- ⇒ schnellere Behandlung von Interrupts
- effizientere Implementierung der Rate der Paketaufnahme

- mittels mmap() Pakete aus dem Benutzerraum vom Puffer lesen, ohne kopieren
- PF_RING bietet die gleiche Programmierschnittstelle wie libpcap an
  ⇒ Legacy support

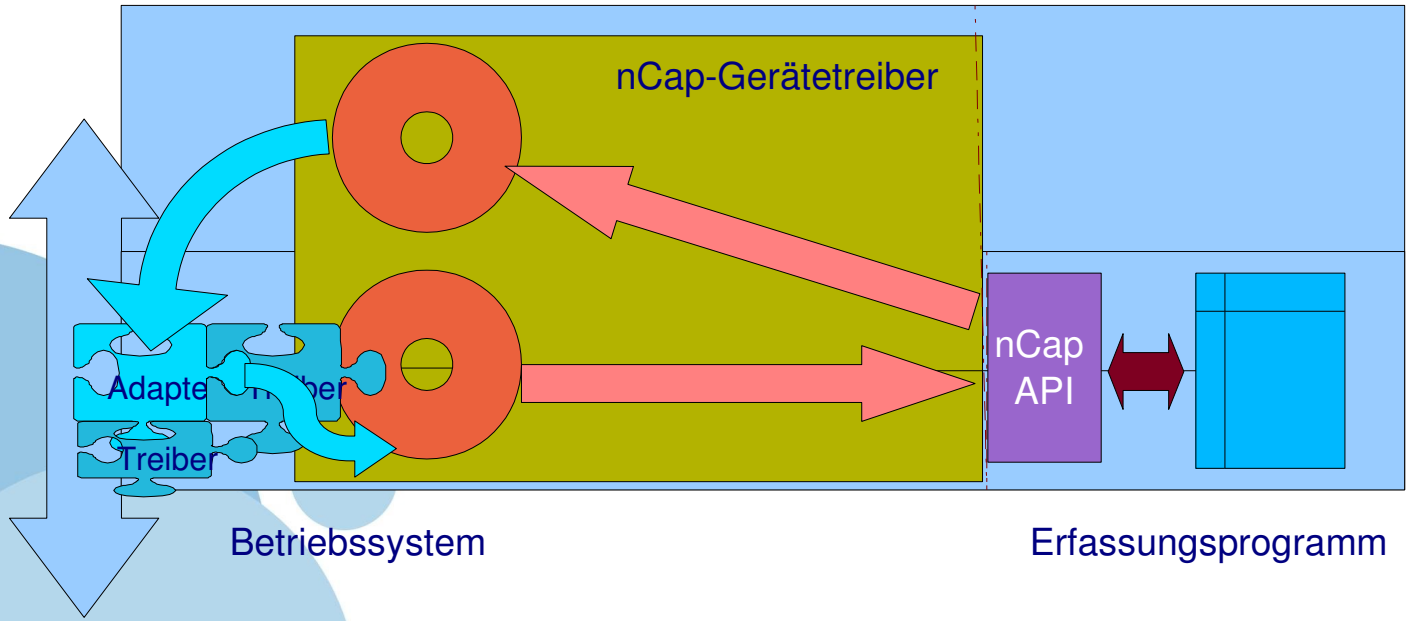
Aus dem Passiven ins Aktiven: nCap

- Hauptproblem Netzwerkverkehr generieren:
  - effizient vom Benutzerraum ins Netzwerk
  - anhand erfassten Pakete
- Herausforderungen:
  - Overhead des Betriebssystems minimieren
  - Latenz zwischen Erfassen und Generieren minimieren

- Anwendung der gewonnenen Erfahrungen aus PF_RING
- Lösung sowohl auf Datenempfang als auch auf Datenversand optimieren

nCap: Aufbau

- Bibliothek im Benutzerraum
- Einführung eines Sendepuffers
- direkte Benutzung des Netzwerkadapters; Puffer mit DMA



nCap: Vorteile

- direkter Zugang ins Netzwerk vom Benutzerraum
- einmaliges Erstellen des Ringpuffers am Anfang, keine Speicherreservierung bei jedem Paket
- Paketversand nur durch Verändern der Leseposition im Netzwerkadapter, Weiterbearbeitung ist Hardware-Implementierung

- Erreichen extrem hoher Senderaten
- Minimierung der Latenz des Sendeprozesses
- Ermöglichung aktiver Messungen und spezifische Netzwerkverkehr-Generierung

nCap vs. PF_RING

- PF_RING : nur passive Paketerfassung,
  - nCap : Senden und Empfangen mit der gleichen Effizienz
- PF_RING und nCap bieten legacy Unterstützung für [libpcap].
- Funktionalität von PF_RING im Kernel,
  - nCap im Kernel und im Benutzerraum
- PF_RING ist ein neues Socket mit normalen Systemaufrufen

- nCap ist eine komplette Lösung mit eigener API ohne Sockets

- für PF_RING Device-Polling vorteilhaft
- für nCap ist nur direkte Steuerung der Netzwerkadapter nötig

nCap vs. PF_RING

- 1 GBit-Netzwerkkonfiguration
- 64 Bytes Paketgröße

Linux 2.4.x Pentium4 1.7GHz CPU PF RING	Linux 2.4.x Pentium III 550 MHz CPU. nCap
205 Mbits	202 Mbits

mit Pentium IV HT erreicht nCap die maximale Geschwindigkeit des GBit-Netzwerkes, ca. 3 mal so schnell wie in diesem Test

Quelle: [Deri nCap]

Zusammenfassung

- Schwächen der heutigen allgemeinen Betriebssysteme bzgl. Paketerfassung.
- Lösung für Interrupt-Live-Lock durch Device-Polling
- Ringpuffer als Basis für die beide Lösungen von Deri
- PF_RING: Kopieren von Paketen durch den Treiber in den Puffer ohne durch die Kernelverwaltung zu wandern

- nCap : Nutzung des Netzwerkkontrollers zum direkten Schreiben in den/ Lesen vom Ringpuffer beim Empfangen/Senden
- PF_RING ist ein Socket
- nCap bietet große Effizienz bei Paketerfassung bzw. Generierung.

Literaturverzeichnis

- [Deri BDP] Luca Deri: Improving Passive Packet Capture: Beyond Device Polling; Proceedings of SANE 2004, 2004.
- [Deri nCap] Luca Deri: nCap: Wire-speed Packet Capture and Transmission; Proceedings of E2EMON, 2005.
- [libpcap] Network Research Group at Lawrence Berkeley Laboratory: libpcap;http://www.tcpdump.org