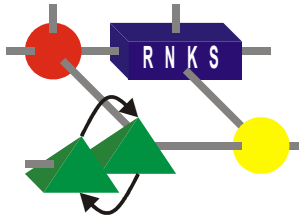
Brandenburgische Technische Universität Cottbus

Lehrstuhl Rechnernetze und Kommunikationssysteme

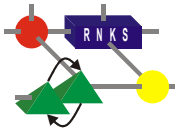


Parallelisierung von NIDS

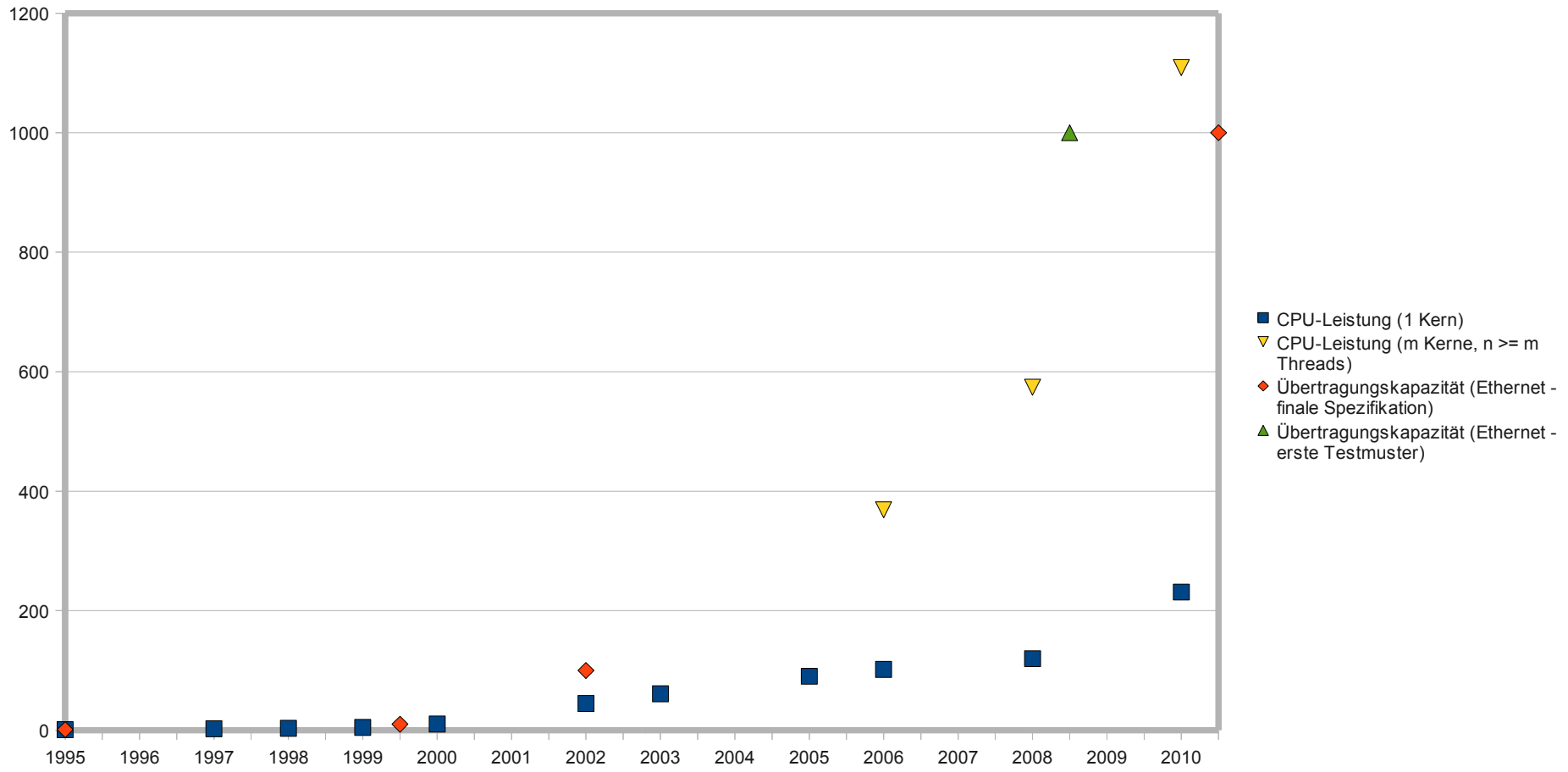
René Rietz

E-Mail: rrietz@informatik.tu-cottbus.de

Inhalt



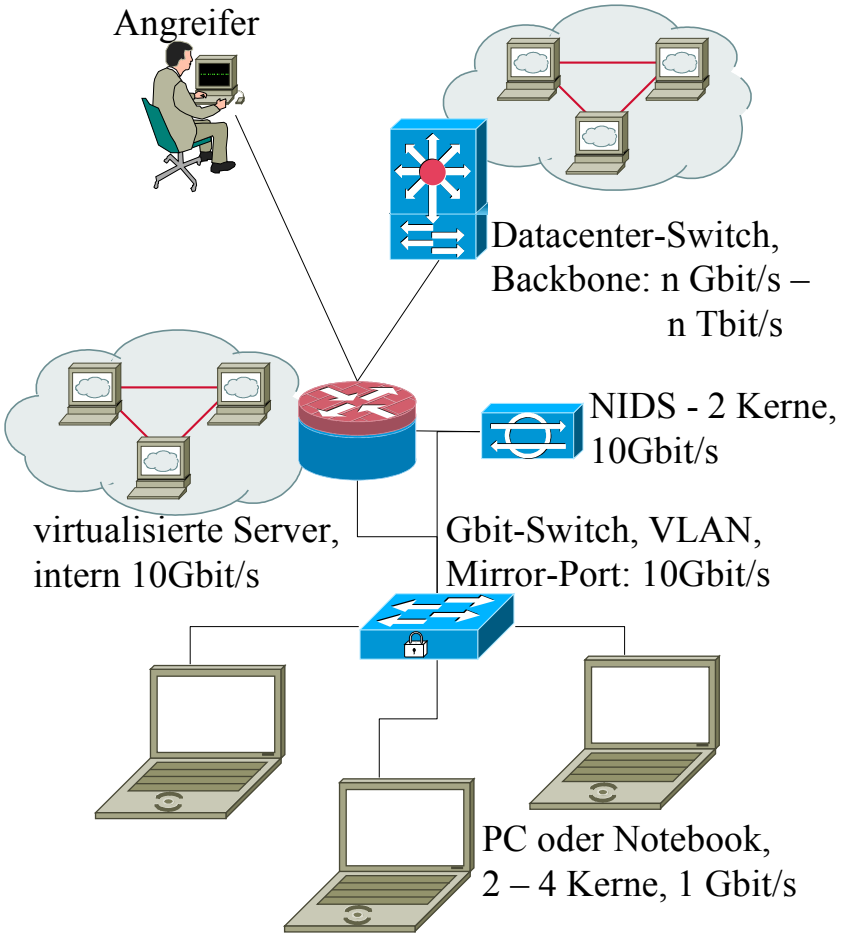
Motivation (CPU-Entwicklung)



Leistungssteigerung CPUs / Steigerung der Ethernet-Übertragungskapazität

Motivation (Komplexität)

- Komplexität des zu untersuchenden Netzverkehrs steigt kontinuierlich
  - Analysekapazität derzeit bis zu 10 Gbit/s (non-x86, angepasste BS)
- weitere Steigerung nur durch Parallelisierung / Verteilung
- Neuimplementierung?
  - Snort: 50 Personenjahre
  - → Anpassung bestehender IDS



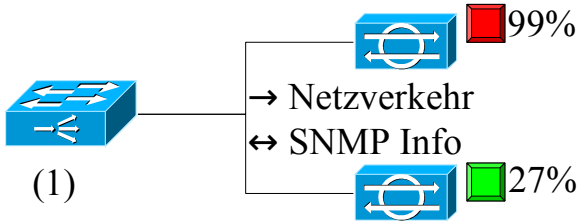
Ansätze zur IDS-Parallelisierung (I)

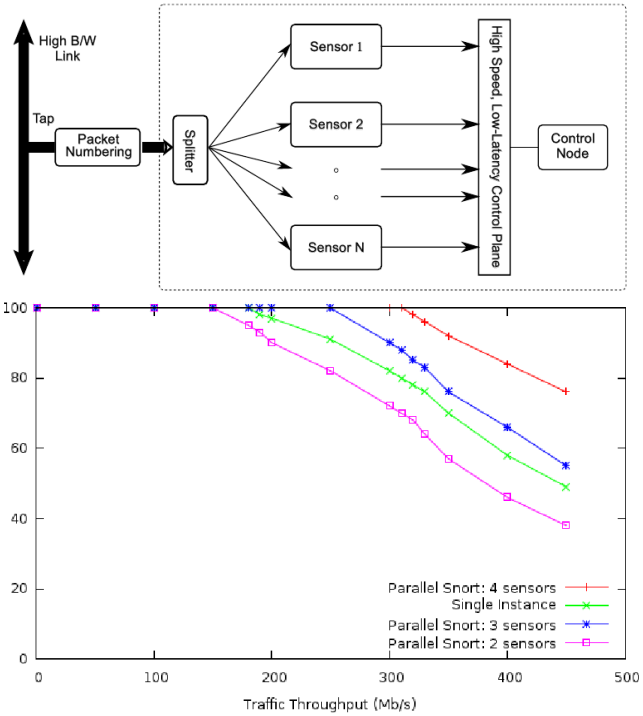
(1) Fluss-basierte Lastbalancierung

- Netzverkehr auf mehrere identisch konfigurierte IDS-Systeme verteilt
- Problem: Analyseaufwand für einzelne Datenströme nicht vorhersagbar

(2) Splitting

- Datenpakete werden round-robin an Sensoren verteilt
- Zustandsinformationen werden zwischen Sensoren synchronisiert





(2) Quelle: Foschini et al. 2008

Ansätze zur IDS-Parallelisierung (II)

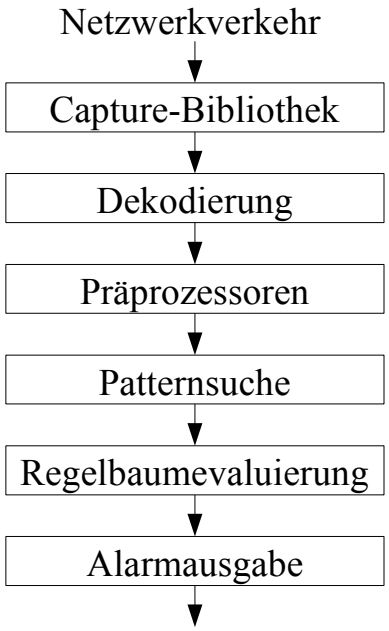
(3) Komponenten-Parallelisierung

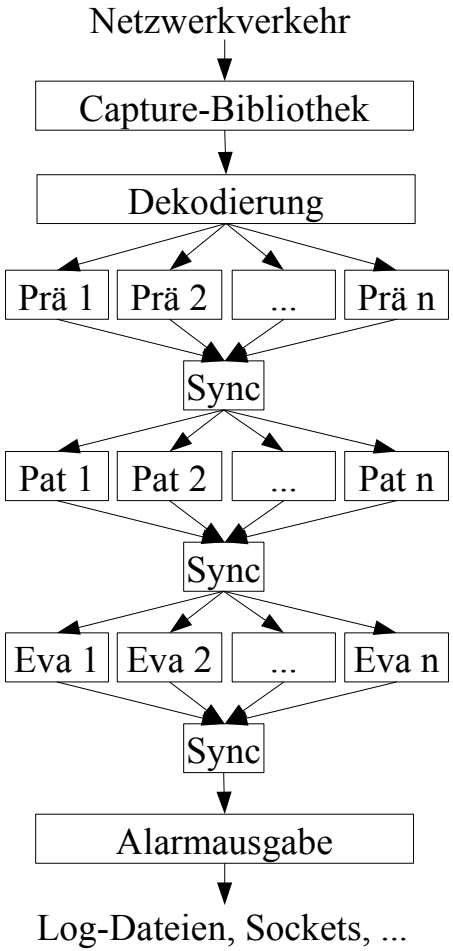
(3.1) Task-parallel

- verschiedene Teilaufgaben parallel
- grobgranular, Leistung häufig nahe bei sequentieller Variante

(3.2) Datenparallel

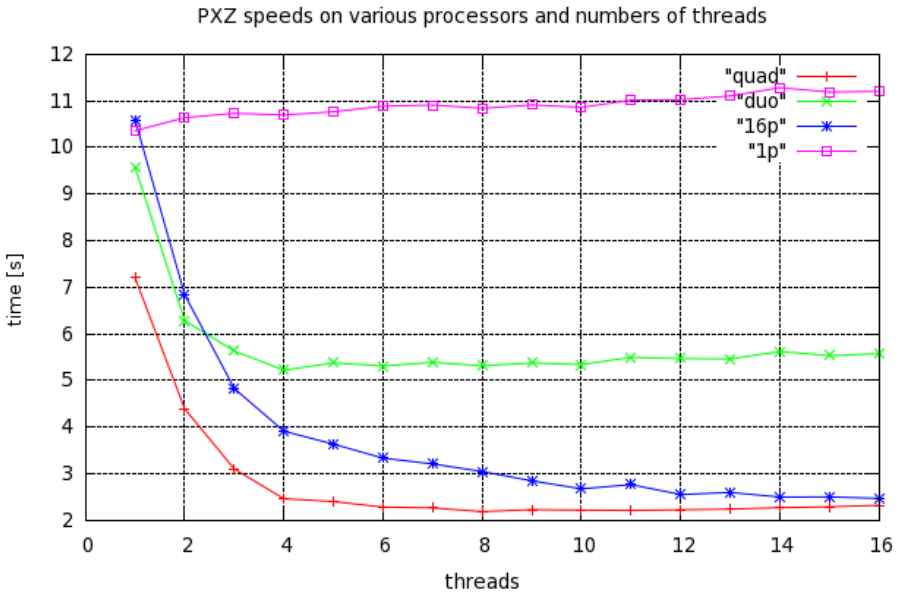
- selbe Aufgabe / disjunkte Daten
- skaliert besser





Einschub: Datenparallele Verarbeitung

- paralleler LZMA-Komprimierer
- Test: Datei 17 MB auf 1, 2, 4, 16 Kernen parallel verarbeitet
  - 1p = P4 3,2 GHz
  - duo = E5200 2,5 GHz
  - quad = Q6600 2,4 GHz
  - 16p = 16 x Intel 3,2 Ghz
- mögliche Probleme: CPU-Cache-Synchronisierung, NUMA, Aufgabe zu klein



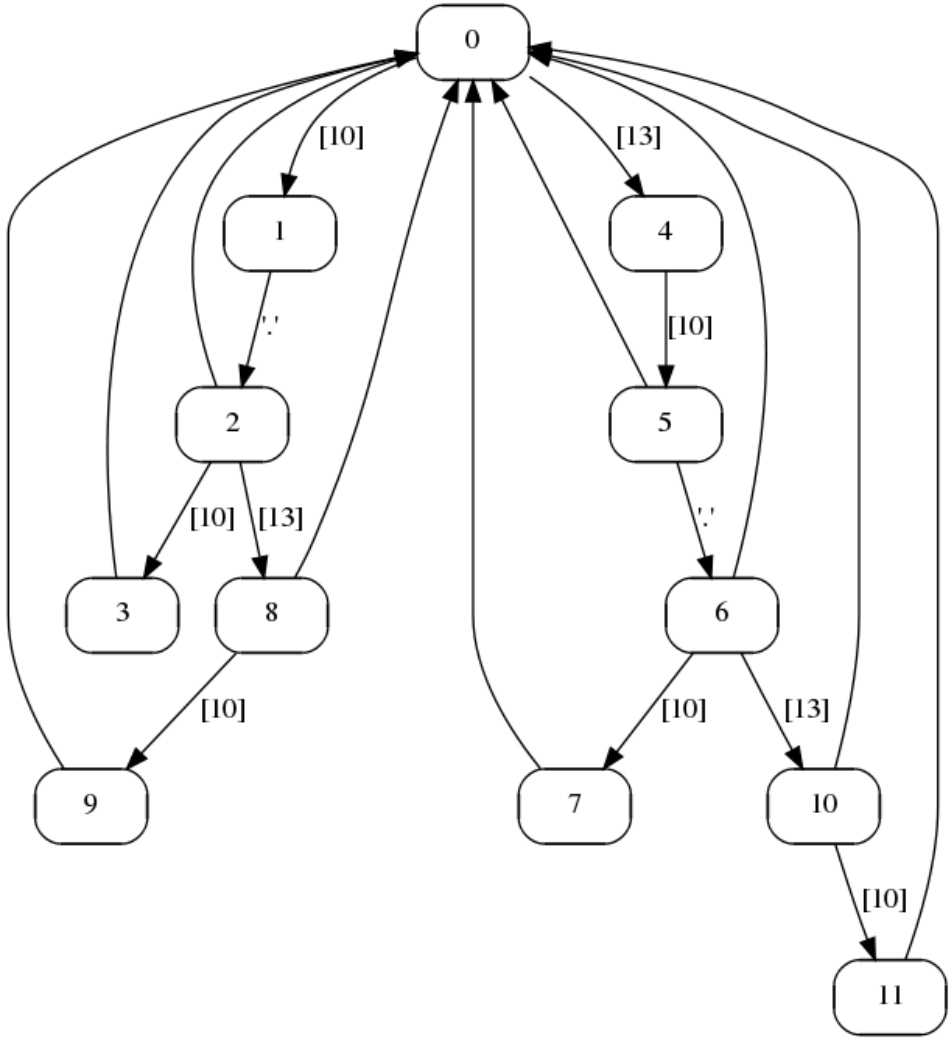
Quelle: http://jnovy.fedorapeople.org/pxz/perf.png

Parallelisierung von Snort (Laufzeit)

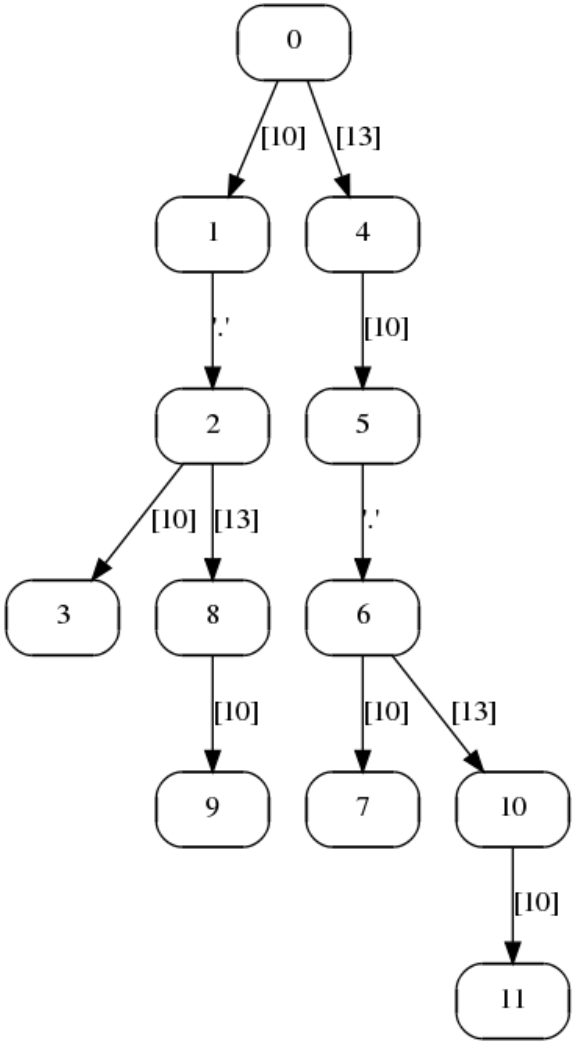
Komponente	Dekodierung	Sitzungsverw.	Patternsuche	Regeleval.
Anzahl Aufrufe	2.668.558	2.487.004	1.221.475	1.123.521
Analysezeit	0,65 s	0,62 s	11,3 s	0,38 s
Analysezeit (Durchschnitt)	0,25 µs	0,25 µs	9,25 µs	0,34 µs
Anteil Gesamtanalyse	3,65%	3,48%	63,48%	2,13%

- 950 MB pcap, gesamte Analysezeit: 17,8 s
- Hauptanteil: Suche von Zeichenketten → Parallelisierung des Suchautomaten
  - Suchautomat arbeitet auf genau einem Paket-, TCP- oder Http-Puffer
- Parallelisierungsmöglichkeiten (geplant):
  - ➔ Suchautomat parallelisieren und auf jeweils einen Puffer anwenden
  - ➔ Suchautomat parallel auf mehrere Puffer/Teilpuffer anwenden
  - – in Erweiterung vorangehende/nachgelagerte Analyseprozesse einbeziehen

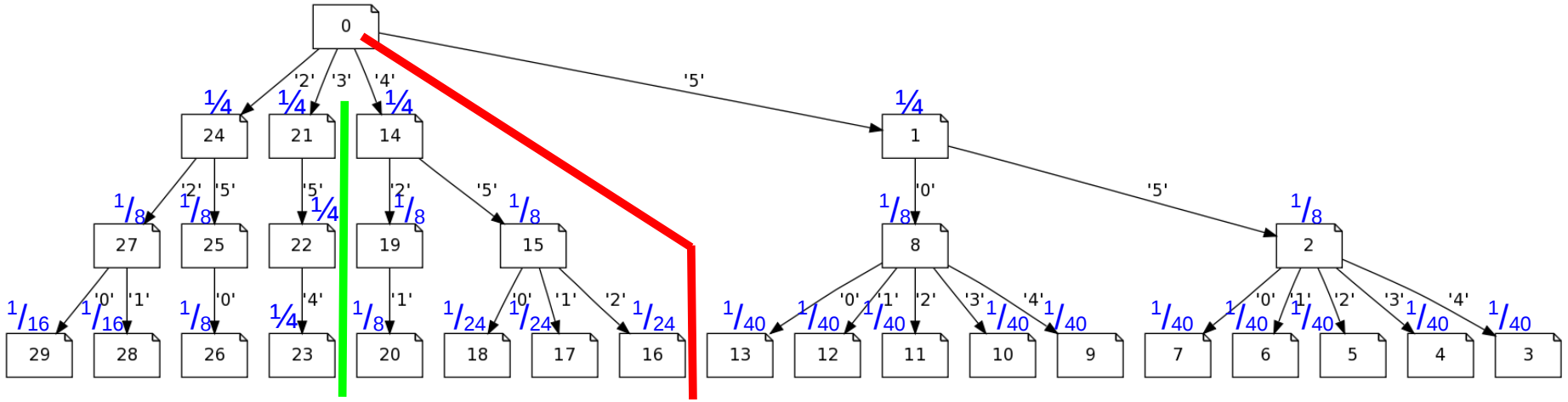
Parallelisierung von Snort (Autom. I)



Parallelisierung von Snort (Autom. II)



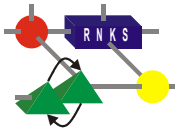
Parallelisierung von Snort (Autom. III)



Thread 1

Thread 2

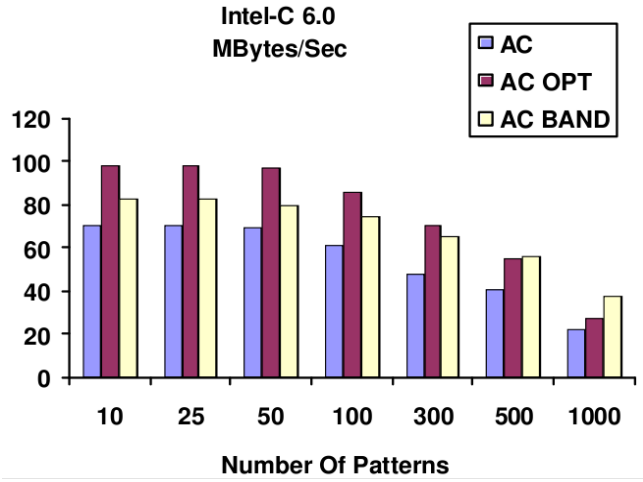
Weitere Forschung



- Verteilung der Analyse
  - Weiterleitung von Verbindungen / Zustandsinformationen in Überlastsituationen
  - Filterung von Netzverkehr, Weiterleitung partieller Verbindungsdaten
  - → dynamische Reaktion auf Ressourcenengpässe bei der Analyse
- Verteilte Erkennung von Angriffen
  - Korrelation von Ereignissen, die in unterschiedlichen Netzbereichen anfallen
- Erkennung verteilter Angriffe
  - DDOS, parallele Wörterbuch-Attacken, verteiltes DNS-Cache-Poisoning?

Parallelisierung von Snort (Cache)

- Rechts: Analyse von Snort, Pentium 4, 1.7 Ghz mit 256KB Cache
- Leistung liegt bei Maximum, wenn Automat vollständig im Cache
- Ansätze zur Parallelisierung, die Automaten duplizieren sind anfällig für Cache-Konflikte
- Parallelisierter Automat sollte Cache-Größe nicht übersteigen



Patterns vs Kbytes

	AC STD	AC OPT	AC BAND
10	58	30	3
25	182	93	9
50	370	188	19
100	729	371	41
300	2333	1160	130
500	3884	1930	239
1000	7567	3760	527

Table 1

Quelle: Marc Norton