

Efficient IP Prefix Lookup Algorithms and Datastructures:

A Framework for Performance Evaluation

Dennis Knorr, thelast@gmx.de
Sommersemester 2006

Motivation

-
- IP Lookup Algorithmen sind der Flaschenhals beim Forwarding
- Wenig direkte Vergleichsmöglichkeiten beim Messen von IP Lookup Verfahren

Herausforderungen

- [□] Longest-Prefix-Matching
- Umstieg von IPv4 auf IPv6
- Classless Interdomain Routing
- Echtzeit

Framework I



- Enthält Algorithmen, Datenstrukturen & Testdaten
- Generische Bitvektor-API einhängbar

Framework II

□

- Das Framework ermöglicht das Benutzen unterschiedlicher Funktionen mit den gleichen Funktionen aus dem Framework
- Daten sind schon im Speicher vorgehalten, sodaß diese nicht von der Festplatte geladen werden müssen

Framework III

- Möglichkeit eigene Testfälle und I/O-Funktionen zu definieren
- Algorithmen können mit minimaler Anpassung für IPv4 und IPv6 genutzt werden

Bitvektor-API

- [□] Generalisierterer Ansatz um mit Bitstrings umzugehen
- Selbstdefinierte Funktionen
- Momentane Implementation auf IPv4 ausgerichtet

Algorithmen

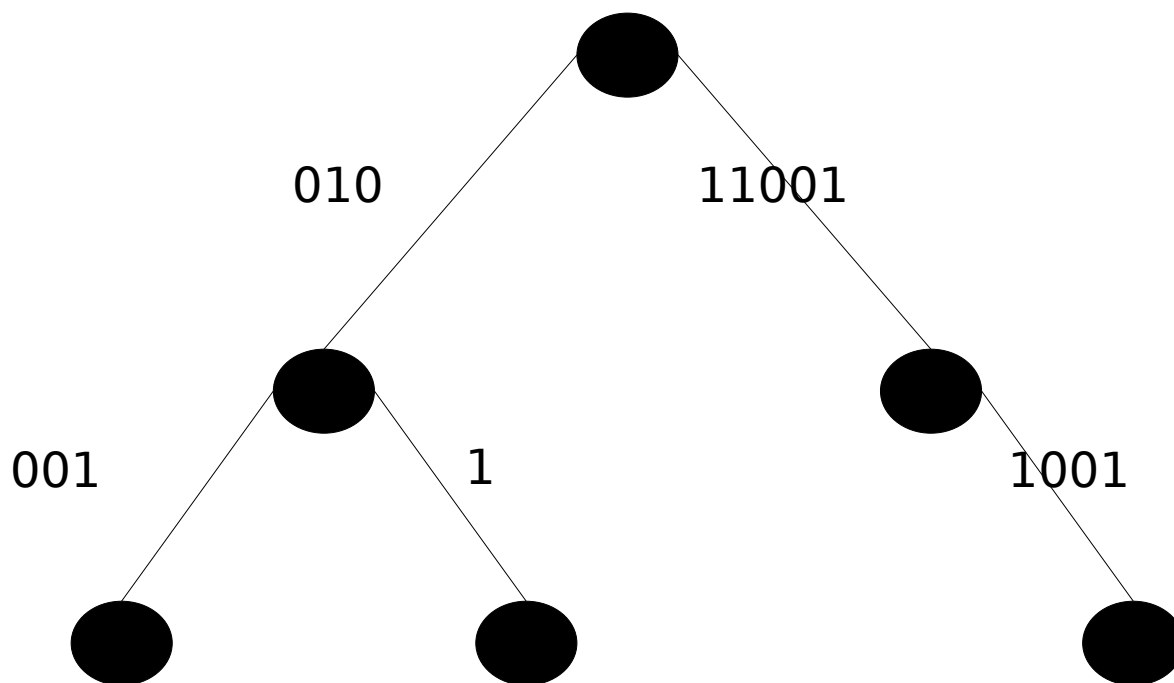
- [□] Bruteforce Ansatz
- Patricia Tree
- Elevator-Stairs-Algorithmus

Bruteforce

- [□] Feld von Präfixen wird angelegt
- Durchsuchen aller Einträge zum Finden des Präfixes
- Keinerlei Optimierungsoperationen

Patricia Tree

□

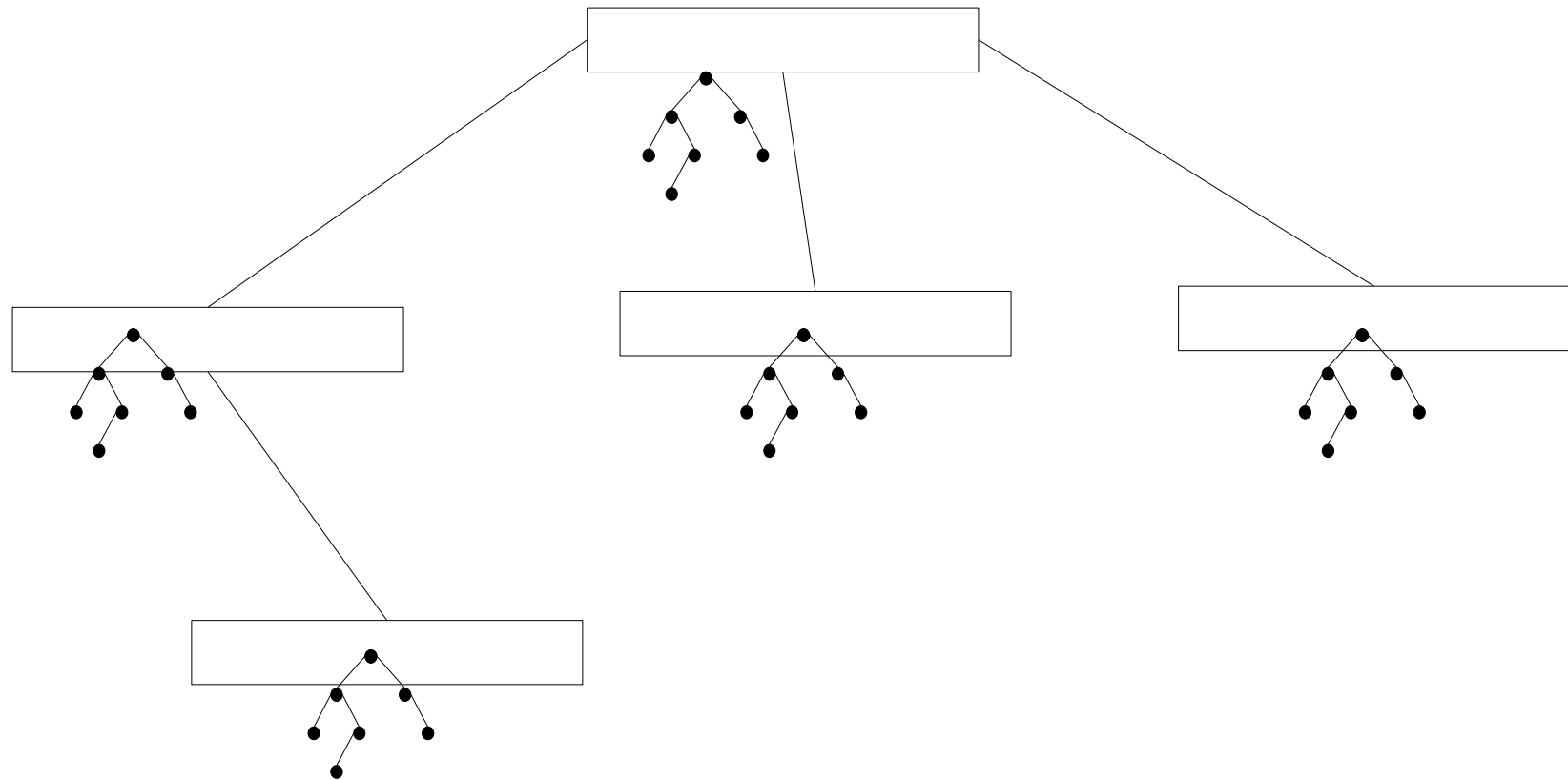


Patricia Tree

- [□] Baum über Bitsequenzen
- Einfüge- und Löschooperationen sind sehr ähnlich zur Suche
- Verbreiteter Ansatz

Elevator-Stairs-Algorithmus

□



Elevator-Stairs-Algorithmus

-
- Datenstruktur aus Wörterbuchbaum und Patricia Tree
- Leistungsgewinn durch konstante Abstände
- Parameter k bestimmt die Abstände

Theoretische Komplexität

Algorithmen	IP Lookup	Insert/Delete	Speicherverbrauch
Feld	$O(NW)$	$O(1)/O(N)$	$O(NW)$
Patricia Tree	$O(W)$	$O(W)$	$O(NW)$
Elevator-Stairs	$O((W/k)+k)$	$O((W/k)+k)$	$O(NW)$

W = Länge einer IP-Adresse

N = Anzahl der Datensätze

Tests

- Vergleich der Effizienz aller Algorithmen
- Exemplarischer Einsatz des Frameworks
- Ermittlung der Testdaten

Testdaten

- Präfixdaten von www.routeviews.org
- Mittels bgpdump extrahiert

- IPs vom Uplink des LRZ
- Mittels ipsumdump extrahiert

Testvorgang

- Für jedes Verfahren ein Programm
- Programme wurden mehrmals aufgerufen, um die Mittelwerte der errechneten Daten zu kalkulieren
- Mittelwerte und Standardabweichung für die Dauer der Funktionen wurden errechnet
- Die Messungen wurde auf einem FreeBSD 6.1 mit einem AMD Athlon 64 3000+ (2 GHz) mit 1GB RAM durchgeführt

Testvorgang II

- Mehrmaliges Ausführen von Insert, Lookup und Delete in den Testfällen für die Patricia Tree und Elevator-Stairs-Algorithmen.
- Bruteforce Prozeduren wurden nur einmal durchgeführt
- 178386 Präfixe wurden eingebunden und NextHops für 200000 IPs gesucht.

Praktische Testergebnisse

Algorithmen	Insert	Lookup	Delete
Bruteforce	23891.00k	0.27k	1.28k
Patricia	928.82k	1326.61k	1066.54k
Elevator-Stairs	450.14k	539.55k	966.43k

- Grössenangaben in Präfix beziehungsweise IP pro Sekunde

Speicherverbrauch der Datenstrukturen

Datenstruktur	Speicherverbrauch
Bruteforce	8027
Patricia Tree	15951
Elevator-Stairs	19770

- Größenangaben in Kilobytes
- Laufzeitgewinn durch mehr Speicher erkauft

Zusammenfassung

- Das Framework erleichtert das Testen von neuen Verfahren
- Elevator-Stairs-Algorithmus hat Potential!
:-)

Ausblick

- Multithreading der Lookupfunktionen
- Echtzeitfähigkeit
- Effizienz von Elevator-Stairs in der Hardware