

## Benchmark-Vergleich verschiedener Datenstrukturen für das IP-Lookup-Problem

### Motivation

Beim Weiterleiten von IP-Paketen in Internet-Routern muss auf Basis der Zieladresse in möglichst kurzer Zeit entschieden werden, an welchen Nachbarknoten im Netz das jeweilige Paket weitergeleitet werden soll. Die zum „Nachschlagen“ verwendete Routertabelle [1] kann durch unterschiedliche Datenstrukturen implementiert werden. Im dynamischen Fall, d.h. wenn auch effizient „Regeln“ für das Weiterleiten hinzugefügt oder entfernt werden sollen, können beispielsweise Strukturen auf der Basis von *Prioritätssuchbäumen* [2,3], *Min-augmented Range Trees* [4] oder *Loser Trees* [5] eingesetzt werden. Diese drei Datenstrukturen sollen vergleichend untersucht werden. Da in der Praxis die tatsächlich benötigte Zeit den Ausschlag gibt, stehen bei dieser Arbeit nicht asymptotische Komplexitätsanalysen, sondern Benchmarks im Vordergrund.

### Zielsetzung

Im Rahmen der Diplomarbeit soll die Performanz der verschiedenen Datenstrukturen durch ein Benchmarking verglichen werden. Dabei sollen die Zeiten sowohl für Suchanfragen als auch für das Einfügen und Entfernen untersucht werden. Es werden ausschließlich die reinen Datenstrukturen (auf isolierten Rechnern) und nicht das tatsächliche Packet Forwarding in echten Netzwerken untersucht.

In einer ersten Phase erfolgt eine Einarbeitung in die Datenstrukturen, die zum großen Teil bereits (in Java) implementiert sind, ggfs. aber noch vervollständigt bzw. angepasst werden müssen. Außerdem ist eine Einarbeitung in den Bereich Benchmarking nötig, um eine methodisch valide Untersuchung zu garantieren.

Auf dieser Basis wird ein Design für die durchzuführenden Testreihen erarbeitet. Der Vergleich erfolgt auf zwei Ebenen: zum einen soll die Anzahl der während einer (Such-, Einfüge-, Entferne-)Operation benötigten Vergleichsoperationen bzw. die Suchpfadlänge (Anzahl der besuchten Knoten) in allen drei Datenstrukturen betrachtet werden. Dieser Test ist unabhängig vom verwendeten Rechnersystem, erfolgt aber dennoch mit Hilfe von Simulationen (bei denen man die Algorithmen „mitzählen“ lässt, was wie oft gemacht wird). Zum anderen sollen dann durch Realzeitmessungen auf einem ausgewiesenen Testrechner die – in diesem Fall von Faktoren wie Compiler, Betriebssystem, Runtime Environment etc. abhängige – tatsächliche Laufzeit (in ms) der Operationen verglichen werden.

### Voraussetzungen:

Erforderlich sind solide Java-Kenntnisse, eine selbständige Arbeitsweise (Einarbeitung in bestehenden, dokumentierten Programmcode und in die angegebene Literatur) sowie Interesse und Spaß am Umgang mit Algorithmen und Datenstrukturen.

### Betreuung:

Prof. Dr. Thomas Ottmann, Lehrstuhl Algorithmen & Datenstrukturen.

Christine Kupich ([kupich@informatik.uni-freiburg.de](mailto:kupich@informatik.uni-freiburg.de)), Tel. 0761 203-8166.

Tobias Lauer ([lauer@informatik.uni-freiburg.de](mailto:lauer@informatik.uni-freiburg.de)), Tel. 0761 203-8170.

**Literatur:**

- [1] Sahni, S., Kim, K.S., and Lu, H. "IP Router Tables". In Mehta, P.M and Sahni, S. (eds), *Handbook of Data Structures and Applications*, Chapman & Hall/CRC, 2005 (chapter 48).
- [2] McCreight, E.M. "Priority search trees". *SIAM Journal on Computing*, 14(2): 257-276, May 1985.
  
- [3] Lu, H. and Sahni, S. "O(log n) Dynamic Router-Tables for Prefixes and Ranges". *IEEE Transactions on Computers* 53(10): 1217-1230, 2004.
  
- [4] Datta, A. and Ottmann, T. "A Note on the IP Table Lookup Problem". Technical Report, December 2004.
  
- [5] Hinze, R. "A simple implementation technique for priority search queues". *Proceedings of the 2001 International Conference on Functional Programming*, Firenze, Italy, September 2001.
  
- [6] de Berg, M., Schwarzkopf, O., van Kreveld, M., and Overmars, M. *Computational Geometry: Algorithms and Applications*. 2nd revised edition 2000.