DoS & Hash Collisions

Steve Ulrich

Micromata GmbH

DoS & Hash Collisions

- Was ist DoS?
- Hashtables – Funktionsweise
- Worst Case
- Wie damit umgehen?

Was ist DoS

- Denial of Service („Dienstverweigerung“)
- Ein Dienst wird so weit Überlastet, dass er zusammenbricht bzw. seinen eigentlichen Dienst nicht mehr verrichten kann.

Warum?

- Erpressung
- Ausschalten von Konkurrenten
- Ausschalten von unliebsamen Angeboten
- Ablenkung
- Berichterstattung ("/."-effect / ge"heise"t)

Wie?

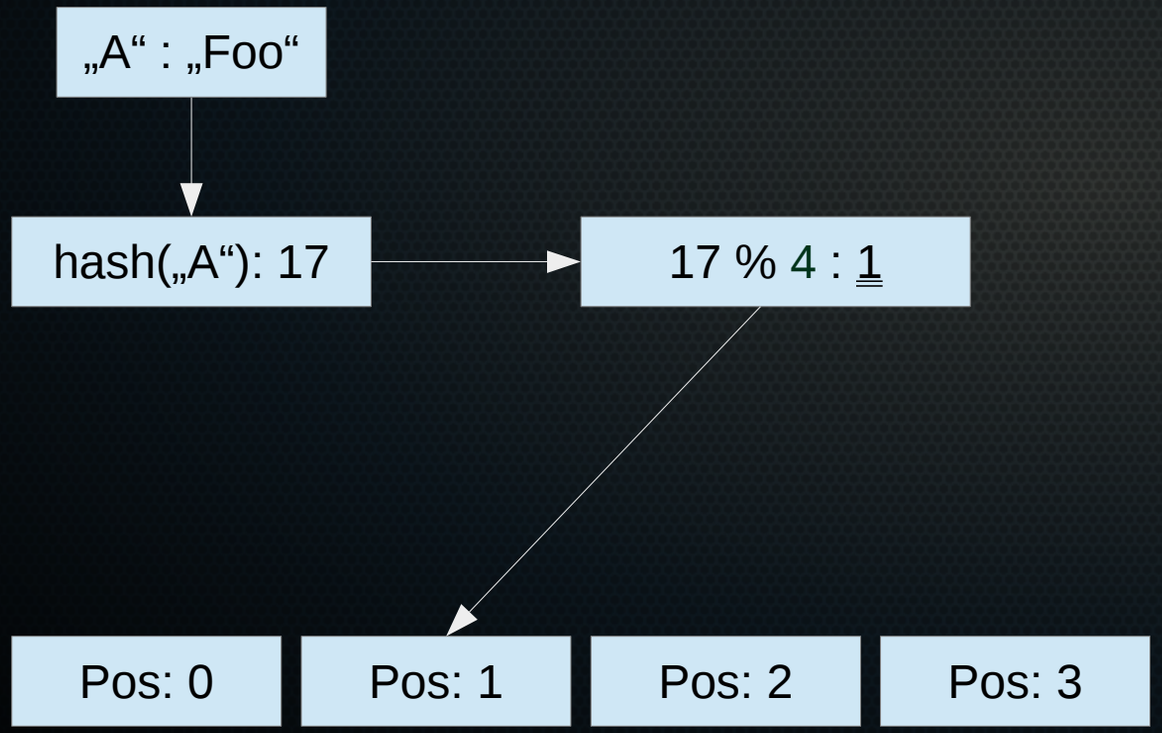
- Überlast durch Anzahl der Zugriffe,
  - Distributed DoS
  - ./-Effekt
- Überlast in der Kommunikation
  - SYN-Flooding
- Überlast durch aufwändige Aufgaben oder Programmfehler
  - Ping of Death
  - Hash Collisions

Hashtables

- Bekannt als: Hash Map, Dictionary, Assoziatives Array
- Charateristik: Key-Value-Datenstruktur
- Vorteile: Im Idealfall sehr schnell, sehr einfach
- Nachteile: Speicherbedarf

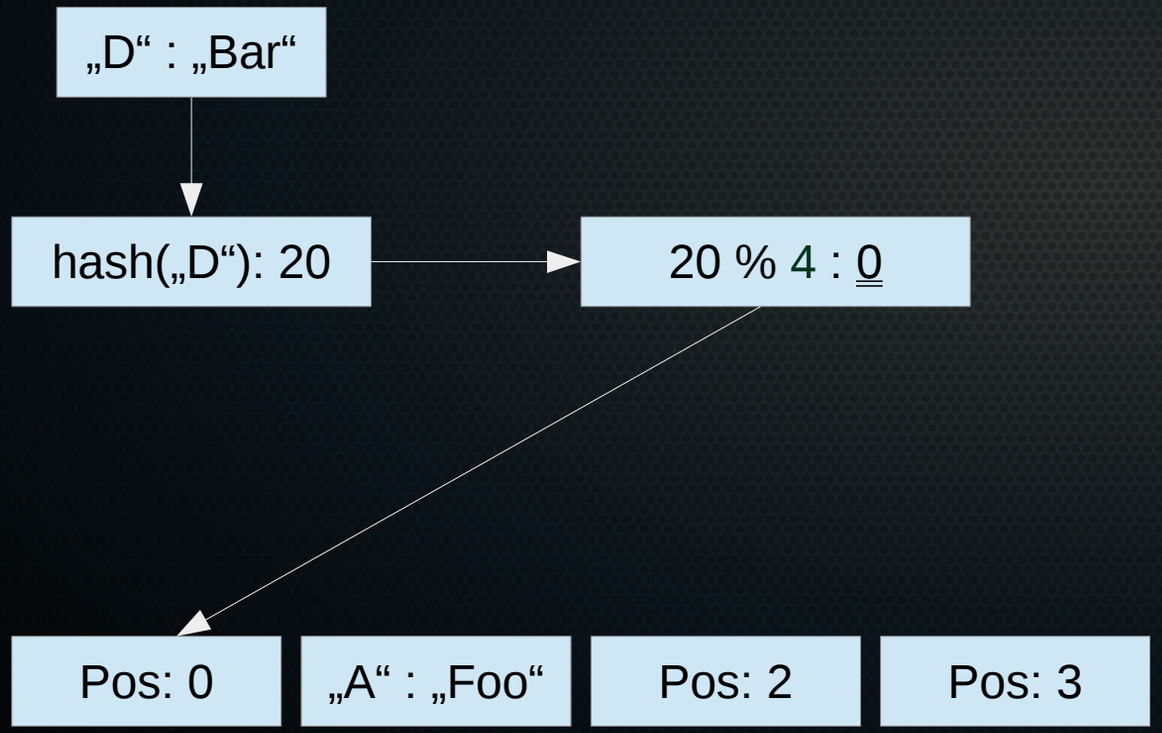
Hashtables

- Größe 4



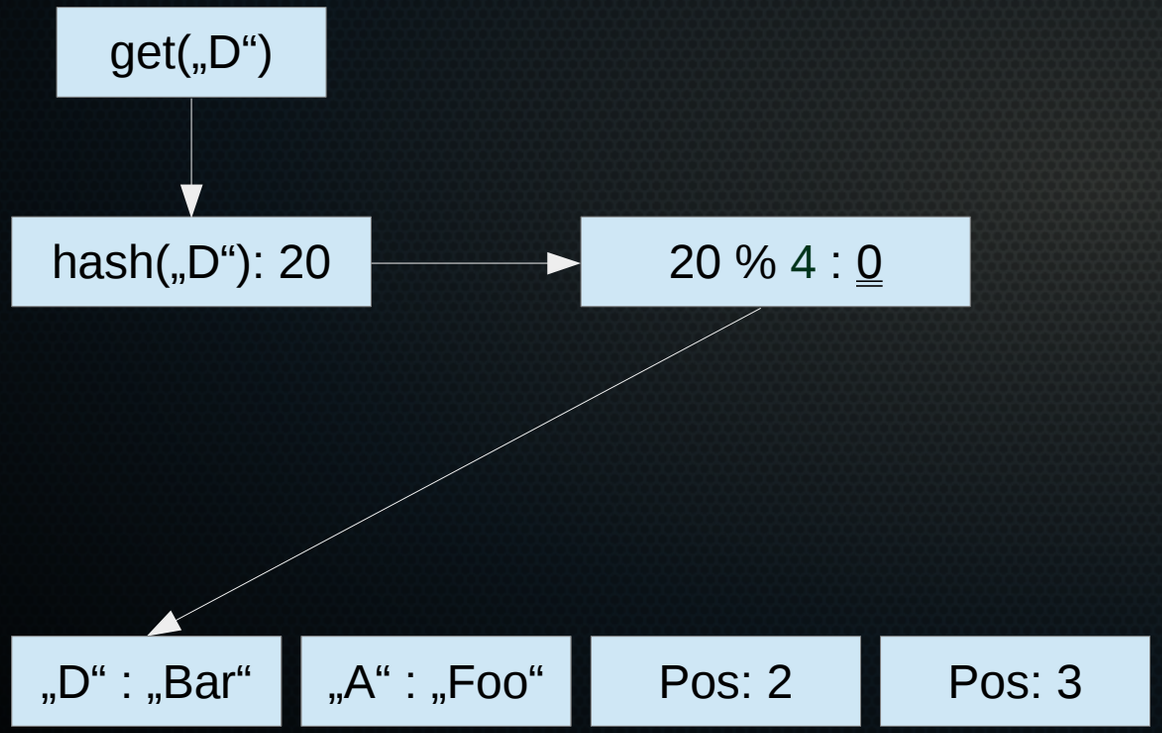
Hashtables

- put in O(1)



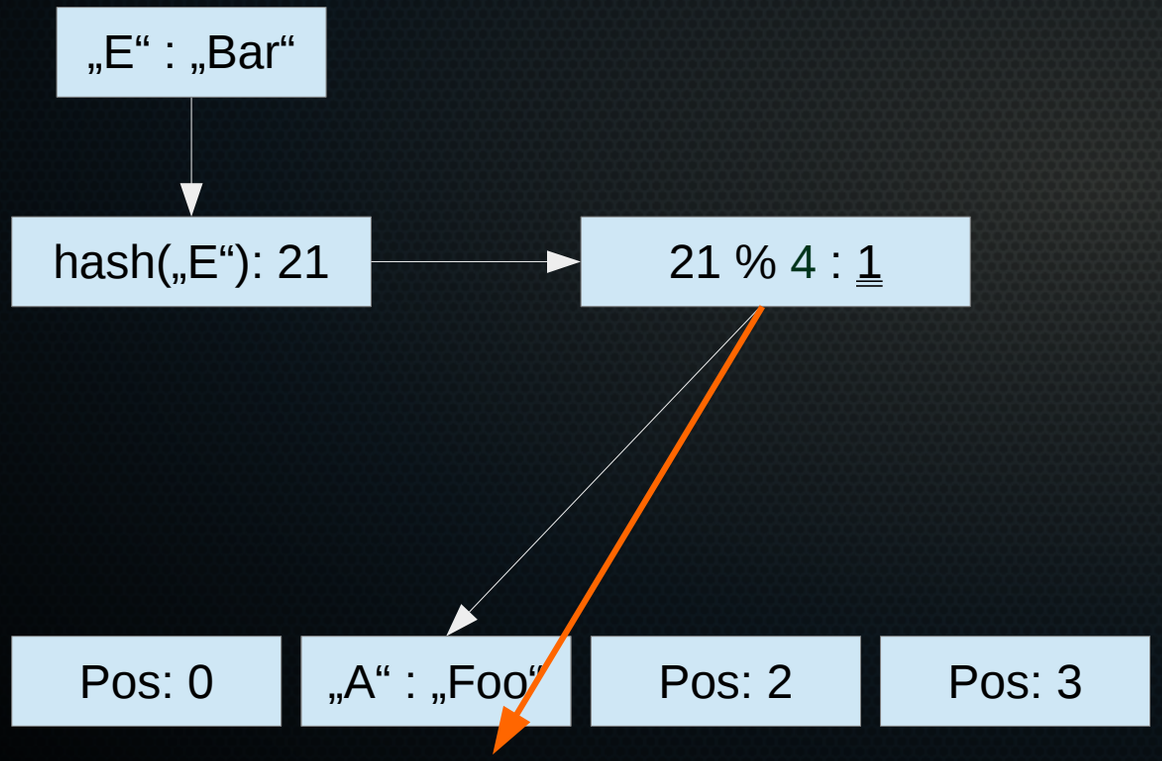
Hashtables

- get(„D“) in O(1)
  - Idealverteilung

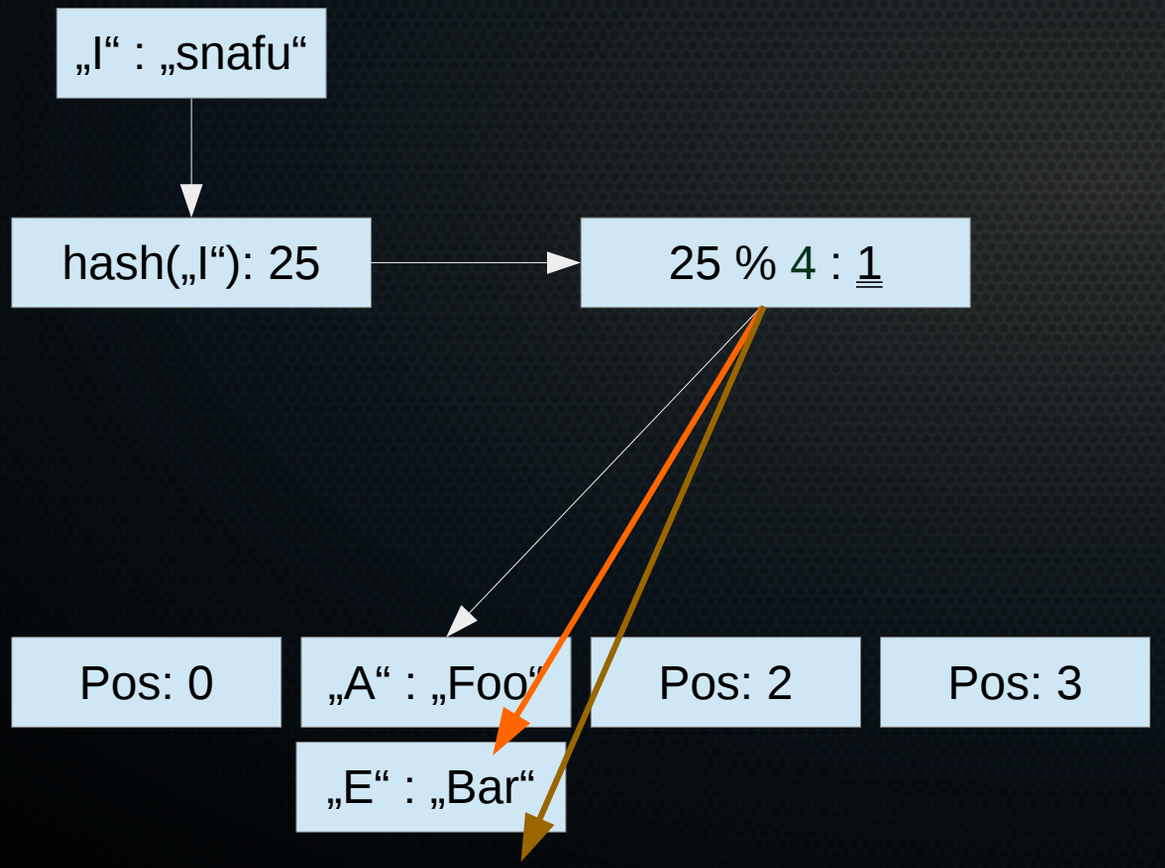


Worst Case

- Key auf Gleicheit prüfen
- Neuer Eintrag



Worst Case



- Key auf Gleicheit prüfen
- Key auf Gleicheit prüfen
- Neuer Eintrag
- put ist O(n)

Worst Case - Angriffsparameter

- Eine Hashtable die ich befüllen kann
- Hashberechnung
- Initialgröße des Arrays
- Vergrößerungen (rehashes)

- Beispiel Tomcat auf Java 7:
  - Http Parameter, Murmur3 32 bit, Größe 16
  - Loadfactor 0.75, scale 2,

Wie damit umgehen?

- Kryptographische Hashes (z.B. SHA)
  - zu Rechenaufwändig
- Seed
  - Time Based Attacks
- Nichtkryptographische Hashes (z.B. MurmurHash)
  - Vorhersagbar
- Nichtkryptographische Hashes + Seed
  - Time Based Attacks

Wie *richtig* damit umgehen?

- Pseudozufällige Funktionen
  - SipHash (z.B. in python)
- Kollisionen zählen
- Andere Strukturen für problematischen Content nutzen