Institut für Informatik

Lehrstuhl für Technische Informatik
Prof. Dr. Björn Scheuermann

Protokolle und Algorithmen WS 2017/18

Übungsblatt 10 [Report 2]

Abgabedatum für die schriftliche Ausarbeitung 31. Januar 2018

Schriftliche Abgaben *von allen* spätestens an diesem Tag via Moodle.

Diese Woche gilt es wieder, Versuche durchzuführen und einen Bericht darüber zu verfassen. Die Rahmenbedingungen sind dieselben wie beim letzten Mal, insbesondere das Arbeiten in Zweiergruppen. Ein Umfang von *etwa fünf Seiten* Text sollte jedoch ausreichen.

Diesmal geht es um die Fairness von und zwischen verschiedenen TCP-Varianten im Linux-Kernel. Zufall kann bei Fairness eine große Rolle spielen. Überlegt euch daher früh, wie ihr dafür Sorge tragt, dass eure Ergebnisse nicht auch zufällig sind; bzw. dass erkennbar ist, wie stark die verbleibenden zufälligen Komponenten im Ergebnis sind und dass diese Ergebnisse aussagekräftig sind (Stichworte: *Wiederholungen*, *Fehlerbalken*).

In der Aufgabenstellung geben wir keine spezifischen Auswertungen oder Darstellungsformen vor. Macht euch deshalb Gedanken, was ihr auswerten wollt, wie das am besten funktioniert und welche Art der Darstellung dafür geeignet ist.

Aufgabenstellung: Euer Ziel ist es, Aussagen darüber zu treffen, wie fair sich zwei TCP-Verbindungen einen Engpass aufteilen. Untersucht dabei mindestens zwei der folgenden Punkte:

- die Fairness zwischen zwei Reno-Verbindungen, (auch) abhängig vom Verhältnis der beiden RTTs,
- die Fairness zwischen zwei CUBIC-Verbindungen, auch hier wieder abhängig vom Verhältnis der RTTs,
- die Fairness zwischen einer Vegas- und einer Reno-Verbindung sowie zwischen einer CUBIC- und einer Reno-Verbindung,
- die Konvergenzgeschwindigkeit zu einer fairen Aufteilung zwischen zwei CUBIC-Verbindungen sowie zwischen zwei Reno-Verbindungen.

Weitere spannende Variationen der Szenarien könnten zum Beispiel viele parallele Verbindungen, Verbindungen ohne ständigen Datenverkehr, Größen von Routerpuffern und schwankende RTTs sein.

Verwendet als Ausgangspunkt die Topologie aus Abbildung 1. Nutzt den Linux-Stack mithilfe der Network-Simulation-Cradle (NSC) und lasst auf Knoten *A* und *B* jeweils eine `OnOffApplication` laufen. Beide Applikationen sollen Daten an eine `PacketSink` auf Knoten *C* senden.

Für einzelne Untersuchungen können die Parameter und ggf. auch die Struktur der Topologie verändert werden. Welche Veränderungen für das Erreichen der Untersuchungsziele genau notwendig bzw. sinnvoll sind, entscheidet ihr selbst.

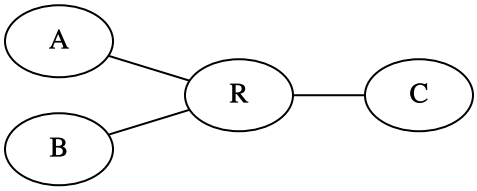

Abbildung 1: Topologie.

Überlegt und erläutert was – ausgehend von der aus der Vorlesung bekannten Theorie – eure Erwartungen sind. Diese Erwartungen sollen dann simulativ überprüft werden. Dabei kann es sich insbesondere lohnen, auch einmal gezielt bestimmte Randfälle (besonders große/kleine BDPs oder sehr große/kleine Puffer oder...) zu betrachten, wenn die Theorie für diese Fälle interessantes Verhalten verspricht.

Hinweise:

- Der Einfachheit halber empfiehlt es sich zwei `PacketSinks` auf dem Knoten C zu installieren.
- Installiert den NSC-Stack nicht auf R, da R mehr als ein Interface hat. Die Installation verläuft analog zu Blatt 8, nun aber mit drei NSC-Knoten (A, B, C).
- Nutzt `Config::Set`, um für einen Konten eine TCP-Variante auszuwählen. Bedenkt dabei (wieder analog zu Blatt 8), dass ihr dies für jeden NSC-Knoten einzeln tun müsst.

  ```
  Config::Set ("/NodeList/0/$ns3::Ns3NscStack<linux2.6.26>/net.ipv4.
      tcp_congestion_control", StringValue ("cubic"));
  ```

  Die hier relevanten Varianten haben die Namen `vegas`, `cubic` und `reno` (für NewReno).
- Ihr werdet viele Simulationen durchführen und die Ergebnisse einsammeln und verrechnen müssen. Macht das nicht alles von Hand, sondern schreibt euch (wenn nicht schon für frühere Aufgaben geschehen) Skripte, die diese Aufgabe so weit wie möglich automatisieren. Das macht es drastisch leichter, zum Beispiel andere Parameterkombinationen auszuprobieren oder nach einem Bugfix die Auswertung mit korrigierten Implementationen zu wiederholen.