Universität
Paderborn
Computer networks
group

Winter term 2017/2018

Mobile communications

Homework assignment 8: Superposition coding,

Due date: 2017-12-13

---

1. **Simulate superposition coding**

   As an **optional** assignment, write a little simulation for superposition coding. Assume two terminals, with different reference SNRs. Do power scaling as discussed in class, overlay the two transmissions. Decode data at the two terminals.

   Think about whether you want to do that simulation on a symbol or a signal level (i.e., whether you just generate complex numbers from the constellation points or whether you want to generate actual sine functions).

2. Discuss the suitability of using CSMA/CD (collision detection) and CSMA/CA (collision avoidance) in Wireless Local Area Networks (WLANs). Provide examples where these methods are inefficient.

   Solution: CSMA/CD: Eine Kollisionserkennung (collision detection) in WLAN ist nicht möglich, weil die Teilnehmer nicht gleichzeitig im drahtlosen Medium hören und übertragen können.

   Dies liegt insbesondere daran, dass wir nur eine Antenne und einen Transceiver haben. CSMA/CA: Kollisionsvermeidung (collision avoidance) durch Wartezeiten beim Sendeversuch: wenn ein freies Medium erkannt wird, kann erst nach einer bestimmten Wartezeit gesendet werden. Wird ein besetztes Medium erkannt, so muss zusätzlich zur vordefinierten Wartezeit noch eine zufällige Zeit (random backoff) gewartet werden, bis wieder auf das Medium zugegriffen werden darf.

   Um Fairheit zu gewährleisten, wird der Backoff-Zähler nach einem verlorenden Medienzugriffsversuch beim nächsten Versuch nicht neu initialisiert sondern von seinem alten Wert weiter heruntergezählt.

   Ineffizienz bei sehr leichter oder sehr starker Last! Dann liegen die zufälligen Backoff-Werte abhängig von Contention Window (CW) sehr eng zusammen oder sehr weit auseinander. => Viele Kollisionen oder unnötig lange Wartezeiten. Das Hidden Station Problem kann durch optionales RTS/CTS gelöst werden. Das Exposed Station Problem kann nicht durch RTS/CTS gelöst werden.

3. **Conflict graphs**

   Suppose we are given a graph V of nodes with edges E. Suppose further we assume the *protocol model*: a node v can only send along edge e to node w if w has no other sending neighbor along E.

   Define the conflict graph $C = (E, E')$: a graph that has the actual communication edges as nodes. Two edges are linked in this graph if they cannot be active at the same time.

   - What does an *independent set* of this conflict graph represent?
   - What does a *maximal independent set* of this conflict graph represent?
   - What does the set of all maximal independent sets represent?
   - How could you interpret a weighted sum of all maximal independent sets?
   - How does that relate to a utilization vector: a vector that specifies, for each edge e in E, the fraction of time it is supposed to be active. Think in terms of schedulability.

   Hint: if you want to read up on this, this paper is a good source: `http://research.microsoft.com/en-us/projects/mesh/interference.pdf`.

   For an encore: Write a simulation to do that! You could follow the following steps.

   - Place nodes on an area (say, a square) as a spatial Poisson process of a certain rate λ. That means:
     - Generate the number of nodes to place as a random variate of a Poisson distribution. (Using a fixed number of nodes is a common fallacy.)
     - For each node, choose x and y coordinates iid from a uniform distribution.
   - Generate the attenuation values based on Euclidean distance between nodes; add a lognormal shadowing component for additional insights.
   - Based on the attenuation values, use a threshold to create the links in the graph. (Think: Which nodes could communicate if there were only noise?)
   - From this graph, compute the conflict graph.
   - Based on the conflict graph, compute sequences of MIS. (Hint: generator expressions should do a nice job here!)
   - Once you have created a new MIS, feed the sequence of generated MIS into an optimizer to find weights among them, optimizing e.g., fairness, throughput, ... or whatever you desire.
   - Check how result improves as you generate more and more MIS.

- What can you say about complexity of this approach?
- Some hints to think about: Graph packages like networkx `https://networkx.github.io`; optimization packages like Pyomo `http://www.pyomo.org`; actual optimizers like Gurobi `http://www.gurobi.com`.

If you want to do that as a Master thesis, talk to me! :-)