On the Reliability of LTE Random Access: Performance Bounds for Machine-to-Machine Burst Resolution Time

Seminar: Distributed Computer Systems SS23

Leonard Kleinberger
lkleinbe@rhrk.uni-kl.de

***Abstract*—tbd**

I. Introduction

- usecases:
  - Power outage
  - Fleet management
  - Sensor networks
  - Emergency situations/environmental desasters

II. Startup Procedure in LTE

- LTE System Setup
  - UE, eNodeB, Core Network
  - We ignore Protocol Stack mostly. We are only interested in Physical Layer to derive the necessary assumptions and from there on only look at message level
  - We are interested in Setup until RRC is connected that is the connection setup between eNodeB and UE From there on the UE can contact the core via eNodeB but that is not in the scope of this work. RRC is will be disconnected when the UE is idle.
- LTE Channels, Timing, Resource Grid
  - Frames/Subframes/Symbols
  - OFDM, multiple carriers
  - Resource Grid
- Over the Air Messages:
  - Reference Signal, PSS, SSS
  - MIB, SIB1, SIB2, more SIBs
  - PRACH Preamble, RA Response, RRC Connection Request, RRC Connection Setup, RRC Connection Setup Complete
  - until PRACH Preamble everything is only downlink -¿ no collisions
  - PRACH Setup is only send in subframes specified by SIB2
  - different Preambles do not collide. They use different sub-carrier. They also do not include any ue specific parameters. Number of available Preambles is specified in SIB2 and depends on cell usage and signal quality
  - RA Response contains the grant in the Resource Grid
  - During RRC Connection Request collision of multiple UEs. eNodeB might decode 1 message or nothing, we cant tell
  - RRC Connection Setup will answer if message decoded successful; connected ue will realize; others will back of
  - UE is successfully connected and responds with RRC Connection setup complete
- Access Barring
  - eNodeB can control how many ues try to connect
  - cellBarred field in sib1 or sib2
  - sib1 for all ues, sib2 for specific types of ues allowing emergency operators to work; Classes go from A to E.
  - Static access barring can reduce the number of admitted ues significantly However statically setting p will slow down connection time when only a few ues are trying to connect
  - dynamic access barring tries to adjust connect probability.
  - optimal algorithm
  - optimal algorithm is not applicable because eNodeB does not know how many UEs try to connect. An Estimation based on failed Preambles is needed.
  - dynamic access barring with estimation

III. Analysis

- Assumptions
  - We only look at number of PRACH contention periods, since the actual number of contention periods is up to the eNodeB
  - RRC connections always fail, if a preamble is chosen by multiple ues. That is essentially the worst case
  - we squash all 4 steps of RA Procedure into 1 Slot. That implies that every ue knows before the next start of the procedure, if the RA procedure before was a success ues dont content in multiple RA Procedures at the same time the eNodeB knows before the next RA Procedure how many ues successfully connected on last try

  - we also assumes that the access barring probability can be updated between every timestep
  - assumptions are pretty generous in my opinion. in reality the Ues only wait between 3-12 subframes before trying to PRACH again. also the change p every step implies we only have 1 contention per 2 frames, essentially limiting this analysis to the configurations with the lowest amount of prach phases.
- QoS, System characteristics, s_i
  - $SC = (M, N, access\_barring\_policy)$
  - we use discrete timesteps to differentiate between the contention periods.
  - at t = 0 N ues wake up
  - $QOS = (b^\epsilon, t, \epsilon)$
  - if a preamble is used multiple times is determined by:

$$s_{i,m} = \begin{cases} 1 & \text{if chosen by 1 UE} \\ 0 & \text{otherwise} \end{cases}$$

    Note that this is essentially an M-channel slotted AlOHA system where each channel corresponds to a chosen preamble
- Iterative Formula
  -

$$B(i+1) = \max\{0, B(i) + a_i - s_i\}$$

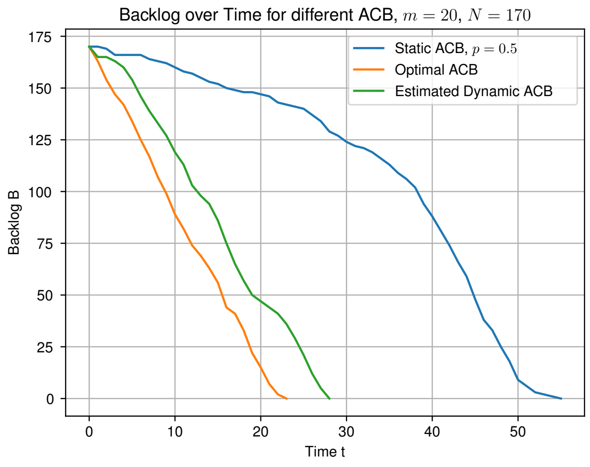
    with

$$s_i = \sum_{m=0}^{M} s_{i,m}$$

  - this already allows analysis in a recursive/iterative way
  - explanation
- Probabilistic Analysis
  - Formel 26
  - first part can be computed fast using formula 25. Which uses formual 24 which uses formula 3
  - second part is relatively slow using iterative formula. however $CM << N$ therefore it is faster than applying iterative formula for all
  - explanation of MGF

IV. Simulation

- Assumptions are pretty much the same as for the analysis. In the Paper the authors used Omnet++ which simulates the Physical Layer aswell For my implementation python is used.
- Implemented Iterative version as iterator
- Pseudocode
- using numpy for simulation to use c++ performance as far as possible. That includes throwing the dice for the access probability (binomial distribution), choosing the preambles (uniform) and counting how often each preamble was selected
- Running large number of simulations



- Multithreading over Searchspace
  Actually its multiprocessing because python interpreter lock which restrics the python interpreter to use only 1 thread at the time to guarantee the consistency of shared objects.

V. Simulation Results

Simulations seem to support analysis. Some configurations are not explicitly stated in the paper.

References