# Assignment 2: Throughput and Interference in LTE

This notebook is best viewed on nbviewer.org.

## 1 Preparation

### What are the parameters for LTE throughput calculation?

The parameters for LTE throughput calculation are the Signal to Noise Ratio (SNR) and the available bandwidth.

### Derive an equation to calculate throughput.

Shannon's equation describes the maximum rate at which information can be correctly received from a channel with a given bandwidth and SNR

In [1]:
```
const shannonsEquation = (snr: number, bandwidth: number) =>
  bandwidth * Math.log2(1 + snr);
```

However, this just is the bound and relies on a few assumptions, like infinite codewords, a random codebook and an AWGN channel.

To calculate the possible throughput for LTE we have to multiply the available bandwidth with the efficiency of the chosen modulation and code rate. These values are chosen based on the CQI according to this table from the specification:

**Table 7.2.3-1: 4-bit CQI Table**

| CQI index | modulation | code rate x 1024 | efficiency |
|---|---|---|---|
| 0 | out of range | | |
| 1 | QPSK | 78 | 0.1523 |
| 2 | QPSK | 120 | 0.2344 |
| 3 | QPSK | 193 | 0.3770 |
| 4 | QPSK | 308 | 0.6016 |
| 5 | QPSK | 449 | 0.8770 |
| 6 | QPSK | 602 | 1.1758 |
| 7 | 16QAM | 378 | 1.4766 |
| 8 | 16QAM | 490 | 1.9141 |
| 9 | 16QAM | 616 | 2.4063 |
| 10 | 64QAM | 466 | 2.7305 |
| 11 | 64QAM | 567 | 3.3223 |
| 12 | 64QAM | 666 | 3.9023 |
| 13 | 64QAM | 772 | 4.5234 |
| 14 | 64QAM | 873 | 5.1152 |
| 15 | 64QAM | 948 | 5.5547 |

In [2]:
```
// Values from the table above
const efficiencyValues = [
  0, 0.1523, 0.2344, 0.377, 0.6016, 0.877, 1.1758, 1.4766, 1.9141, 2.4063,
  2.7305, 3.3223, 3.9023, 4.5234, 5.1152, 5.5547,
];

const getEfficiency = (cqi: number) => {
  const efficiency = efficiencyValues[cqi];
  if (efficiency === undefined) {
    throw new Error("CQI must be an integer between 0 and 15");
  }
  return efficiency;
};
```

NOTE: There are also other tables, but we choose this one.

To index the table we need the CQI value. The CQI is based on the SNR, we use the the function from the exercise sheet to convert them.

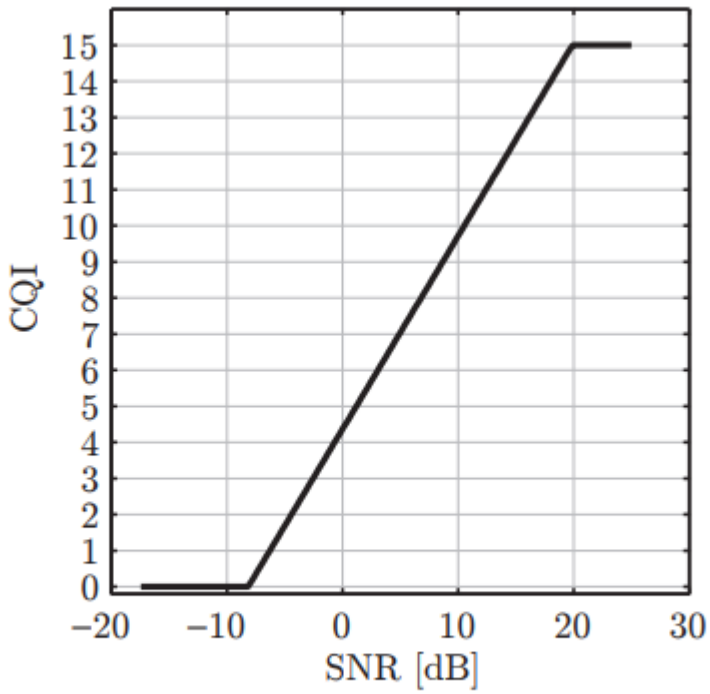



Figure 1: A model for SNR-to-CQI mapping from [4] for a Block Error Rate (BLER) of 10%.

In [3]:
```
const snrToCqi = (snrDb: number) => {
  const cqi = snrDb * 0.525 + 4.5;
  const clampedCqi = Math.max(0, Math.min(15, cqi));
  const clampedAndRoundedCqi = Math.round(clampedCqi);
  return clampedAndRoundedCqi;
};
```

When we combine these two, we can calculate the throughput in based on the SNR and the bandwidth for LTE

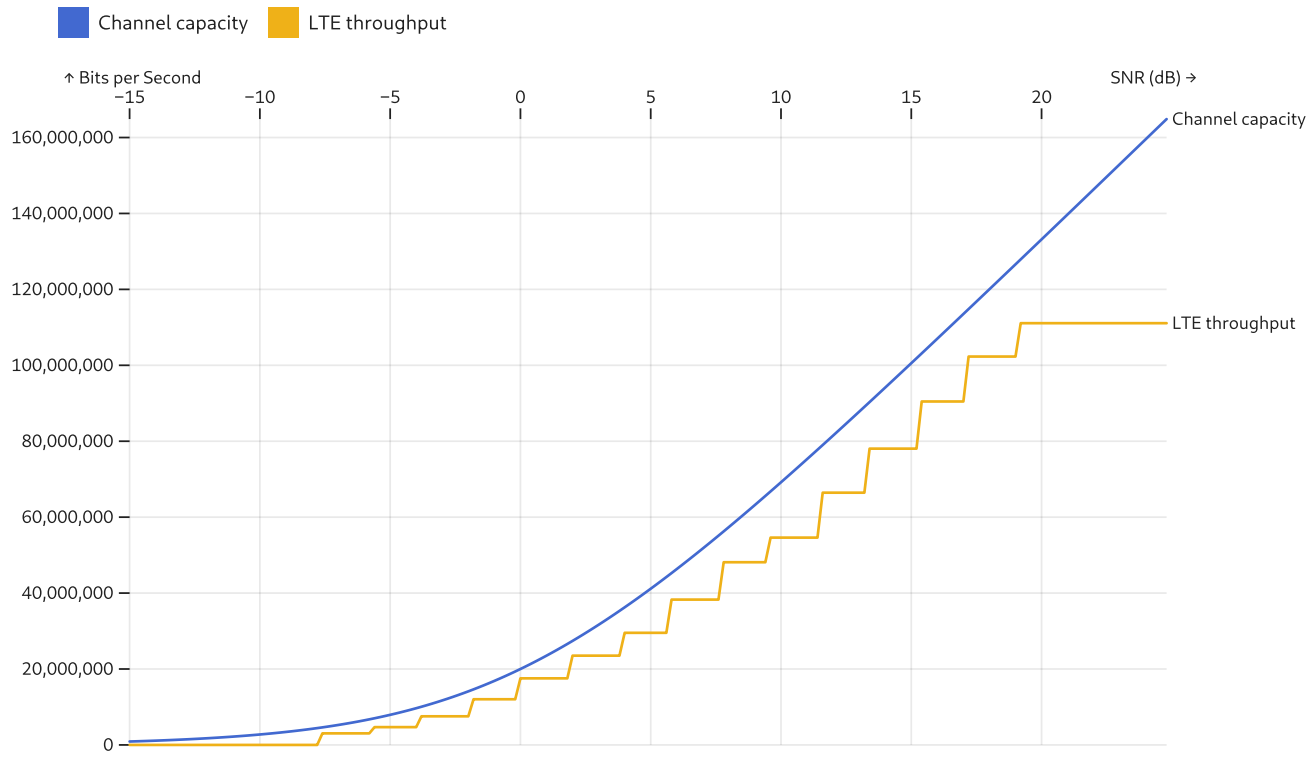
In [4]:
```
const lteThroughput = (snrDb: number, bandwidth: number) =>
  bandwidth * getEfficiency(snrToCqi(snrDb));

console.log(
  `The bandwidth for 20mHz and a SNR of 7dB is ${lteThroughput(
    7,
    20e6
  )} bits per second`
);
```

```
The bandwidth for 20mHz and a SNR of 7dB is 38282000 bits per second
```

If we compare that function with the channel capacity we can see that it is always slightly lower:

In [5]:
```ts
import { plotFunctions, plotBoxes, plotBars, plotStackedBars } from "./helper.ts";

const dbToFactor = (db: number) => Math.pow(10, db / 10);

const factorToDb = (factor: number) => 10 * Math.log10(factor);

const bandwidth = 20e6;

await plotFunctions({
  from: -15,
  to: 25,
  step: 0.2,
  xName: "SNR (dB)",
  yName: "Bits per Second",
  colorName: "__color",
  data: [
    [
      "Channel capacity",
      (snrDb) => shannonsEquation(dbToFactor(snrDb), bandwidth),
    ],
    ["LTE throughput", (snrDb) => lteThroughput(snrDb, bandwidth)],
  ],
});
```

Out[5]:




# 2 LTE throughput at the link layer

## Measurements

We used a Samsung Galaxy S10e to take our measurements. We measured the bandwidth and SNR five times on the same Smartphone and got the following results:

In [6]:
```ts
const measurementsExerciseTwo = [
  { snrDb: 8.4, bandwidth: 20e6 },
```

```
  { snrDb: 11.8, bandwidth: 10e6 },
  { snrDb: 8.6, bandwidth: 20e6 },
  { snrDb: 12.4, bandwidth: 10e6 },
  { snrDb: 7.4, bandwidth: 20e6 },
];
```

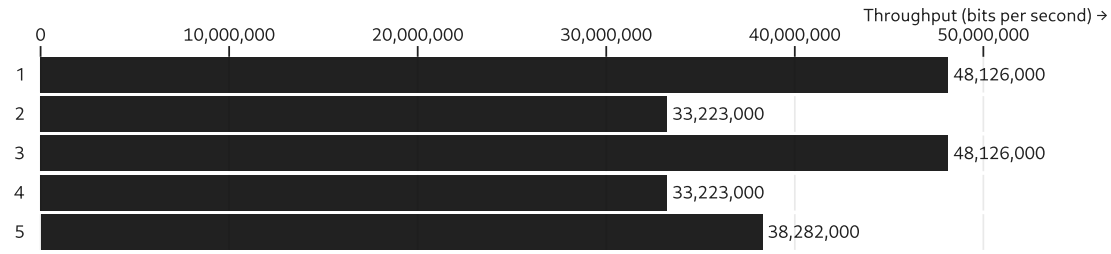
Our phone sometimes uses 20mHz nd sometimes 10mHz bandwidth. When using 10mHz the SNR is a lot higher than when using 20mHz. I would assume that our phone chooses the connection with the better datarate, so the SNR probably compensates for the lower bandwidth.

## Calculating the link-layer throughput

To calculate the link-layer throughput we inserted our values into the `lteThroughput` function we defined above. The chart below shows the calculated values.

In [7]:
```
const linkLayerThroughput = measurementsExerciseTwo.map(
  ({ snrDb, bandwidth }) => lteThroughput(snrDb, bandwidth)
);

plotBars({
  data: linkLayerThroughput,
  yName: "Throughput (bits per second)",
});
```

Out[7]:




## Comparing the results

As we only did a single measurement we can not really compare our results to anything. The following box plot shows our measurement. It has no whiskers, because the IQR is identical to the full range.

In [8]:
```
plotBoxes({
  data: [["Indoors", [linkLayerThroughput]]],
  yName: "Datarate (bits per second)",
});
```

Out[8]:

# 3 Interference in LTE

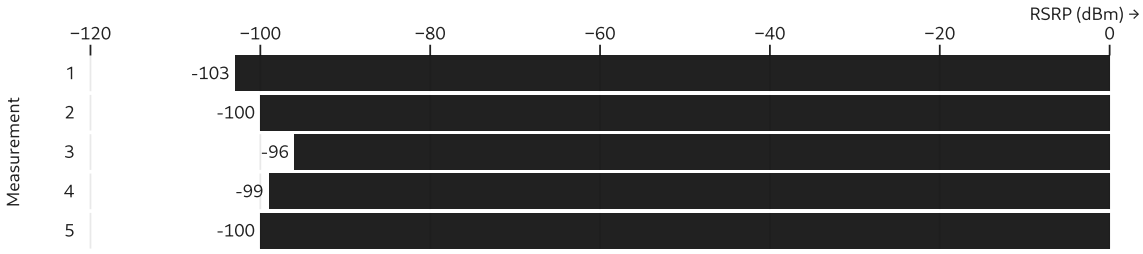
## Measurements

For the third exercise we measured the RSRP and the SNR of the connected cell, and the RSRP of at least three neighbouring cells. The basic setup is same same as previously. For better comparability we discarded all measurements where the bandwidth was only 10mHz. In total we made the following 5 measurements:

In [9]:
```
const measurementsExerciseThree = [
  {
    rsrpDbm: -103,
    snrDb: 7.6,
    neighbours: [
      { id: 51, rsrpDbm: -112 },
      { id: 53, rsrpDbm: -108 },
      { id: 485, rsrpDbm: -111 },
    ],
  },
  {
    rsrpDbm: -100,
    snrDb: 8.2,
    neighbours: [
      { id: 51, rsrpDbm: -113 },
      { id: 53, rsrpDbm: -115 },
      { id: 485, rsrpDbm: -113 },
      // {id: 331, rsrpDbm: -114},
    ],
  },
  {
    rsrpDbm: -96,
    snrDb: 9.8,
    neighbours: [
      { id: 326, rsrpDbm: -108 },
      { id: 155, rsrpDbm: -109 },
      { id: 429, rsrpDbm: -103 },
    ],
  },
  {
    rsrpDbm: -99,
    snrDb: 8.4,
    neighbours: [
      { id: 51, rsrpDbm: -109 },
      { id: 485, rsrpDbm: -113 },
      { id: 485, rsrpDbm: -113 }, // There were only two neighbours, so we copied the
    ],
  },
  {
    rsrpDbm: -100,
    snrDb: 7.2,
    neighbours: [
      { id: 51, rsrpDbm: -113 },
      { id: 53, rsrpDbm: -112 },
      { id: 253, rsrpDbm: -113 },
    ],
  },
];
type Measurement = (typeof measurementsExerciseThree)[number];

plotBars({
  data: measurementsExerciseThree,
  yName: ["rsrpDbm", "RSRP (dBm)"],
```

```
  xName: "Measurement",
});
```

Out[9]:




## Calculating the SINR

The Signal-to-Interference-plus-Noise Ratio (SINR) is calculated from the measured RSRP of the connected and the neighbouring cells. The value for noise was specified as -134 dBm in the assignment sheet. I am not sure if this is really a constant. We could probably try to calculate it from the SNR and the RSRP.

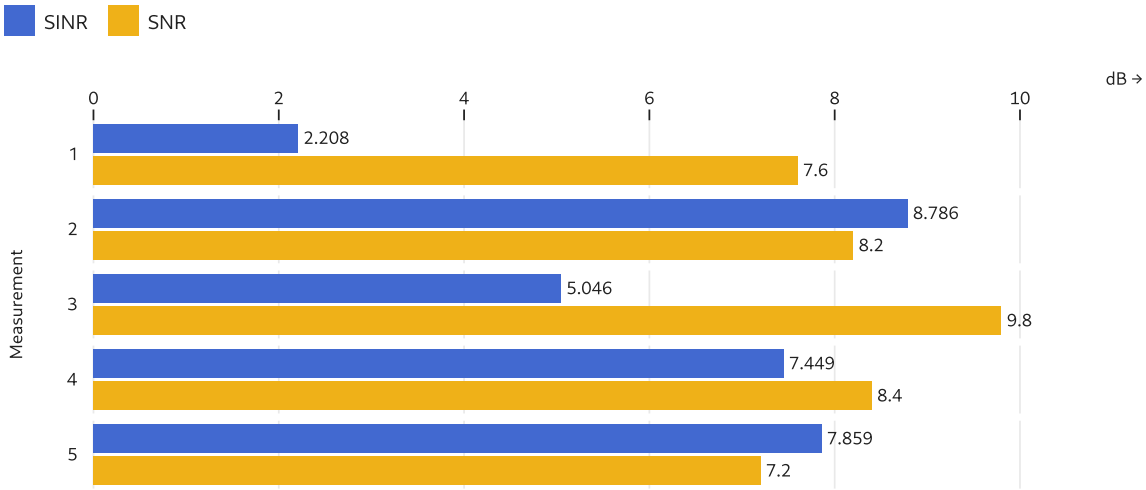
We used the following function to calculate the SINR:

In [10]:
```
const noise = dbToFactor(-134);

const calculateSinr = ({ rsrpDbm, neighbours }: Measurement) =>
  rsrpDbm -
  factorToDb(
    noise +
      neighbours.reduce(
        (acc, neighbour) => acc + dbToFactor(neighbour.rsrpDbm),
        0
      )
  );
```

## Comparing the calculated SINR to the SNR

In [11]:
```
plotBars({
  data: [
    ["SINR", measurementsExerciseThree.map(calculateSinr)],
    ["SNR", measurementsExerciseThree.map((x) => x.snrDb)],
  ],
  yName: "dB",
  xName: "Measurement",
});
```

Out[11]:

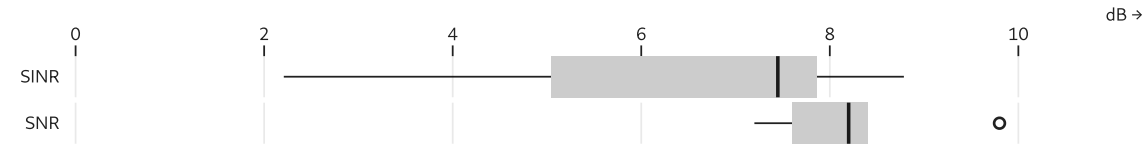



The chart above shows the calculated SINR and the measured SNR. For measurements 2, 4, and 5 the SINR and SNR are quite close. For measurement 1 and 3 the SNR is significantly better than the SINR. I am not sure why this is. I would have assumed that the SINR is always worse than the SNR as it is compares the signal strength to the noise plus the interference. This is however not always the case in our measurements.

In [12]:

```
const sinrMeasurements = measurementsExerciseThree.map(calculateSinr)
const snrMeasurements = measurementsExerciseThree.map((x) => x.snrDb)

plotBoxes({
  data: [[
    ["SINR", sinrMeasurements],
    ["SNR", snrMeasurements],
  ]],
  yName: "dB"
});
```

Out[12]:

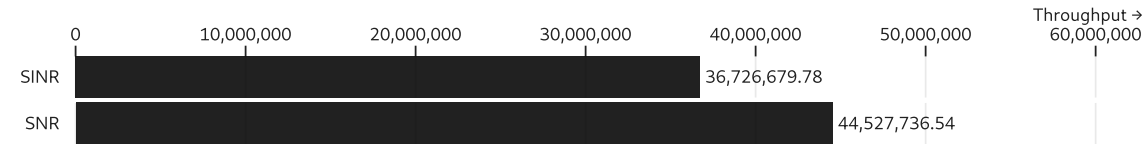



## Comparing throughput

In [13]:

```
plotBars({
  data: [
    [
      ["SINR", sinrMeasurements.map((sinr) => lteThroughput(sinr, 19.579e6))],
      ["SNR", snrMeasurements.map((snr) => lteThroughput(snr, 19.579e6))],
    ],
  ],
  yName: "Throughput",
});
```

Out[13]:



The throughput calculated we calculated with the SINR is lower than the throughput calculated with the SNR.

In [14]:
```
const average= (data: number[]) => data.reduce((acc, x) => acc + x, 0) / data.length
const averageSnrThroughput = average(snrMeasurements.map((snr) => lteThroughput(snr,
const averageSinrThroughput = average(sinrMeasurements.map((sinr) => lteThroughput(si
const lossDueToInterference = averageSnrThroughput - averageSinrThroughput

console.log(`Due to interference we lose ${(lossDueToInterference/1e6).toFixed(2)} MB
```

```
Due to interference we lose 7.80 MBit/s
```