Tomography

In [71]:
```
rm(list=ls())
library(pracma)
```

In this lab, we will study:

- How to simulate a network with packet loss;
- How to estimate the packet loss per link using end-to-end measurements;
- How to allocate the probes in a network in order to maximize the Fisher Information.

Simulation model

We assume the following model:

- Let $G = (V, L)$ a network with nodes V and links L;
- Each link $l \in L$ is associated with a success rate θ_l. In other words, the probability that a link l succeeds to transmit a packet is equal to θ_l. Let $\theta := [\theta_l]_{1 \leq l \leq |L|}$
- Let P be the set of probing paths in G;
- The observation matrix A is a $|P| \times |L|$ matrix where $A_{pl} = 1$ iff link l is on path p,
- Let n_p be the number of probes sent through path p and $n := [n_p]_{1 \leq p \leq |P|}$ is the associated vector of dimension $|P|$.

1/ Complete x in the function *simulation(A,Theta,n)*. x is a vector. The p-th element of x is equal to the number of packets that have been successfully transmitted through path p. (Hint: Use rbinom(1,1,q), where q is the probability for the i-th probe to be successfully transmitted through the path p).

In [72]:
```
Simulation <- function(A,Theta,n) {
    x<-array(0,c(length(n)))
    for (p in 1:length(n)) {
        q <- prod(Theta[A[p,] == 1])
        x[p] <- sum(rbinom(n[p], 1, q))
    }
    return(x)
}
```

2/ Define the observation matrix A such that

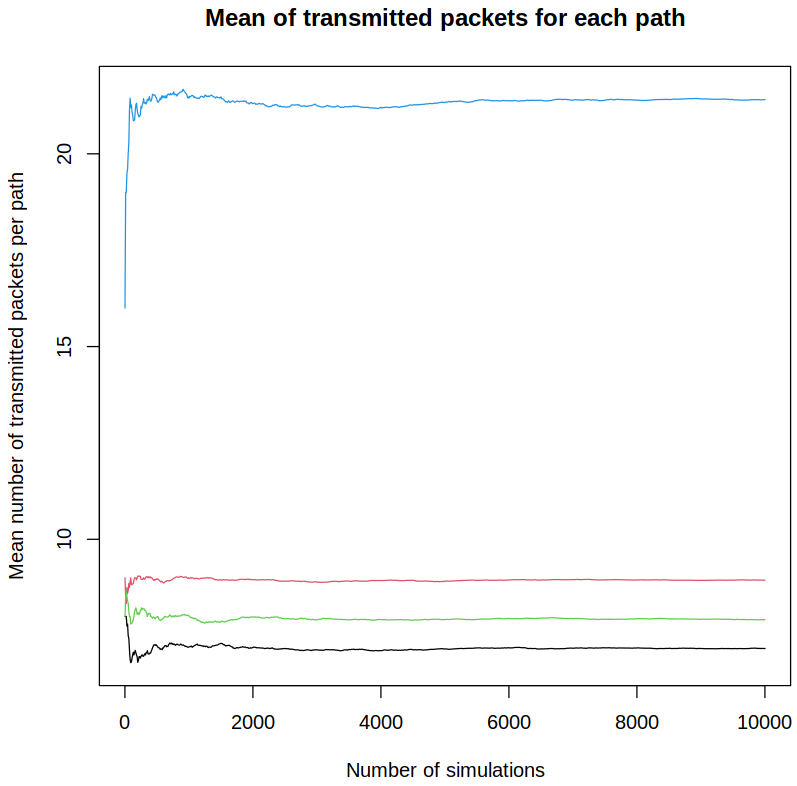
$$A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix},$$

the vector $n.experience = [10, 10, 10, 30]$ and finally $Theta = [0.8, 0.9, 0.99]$.

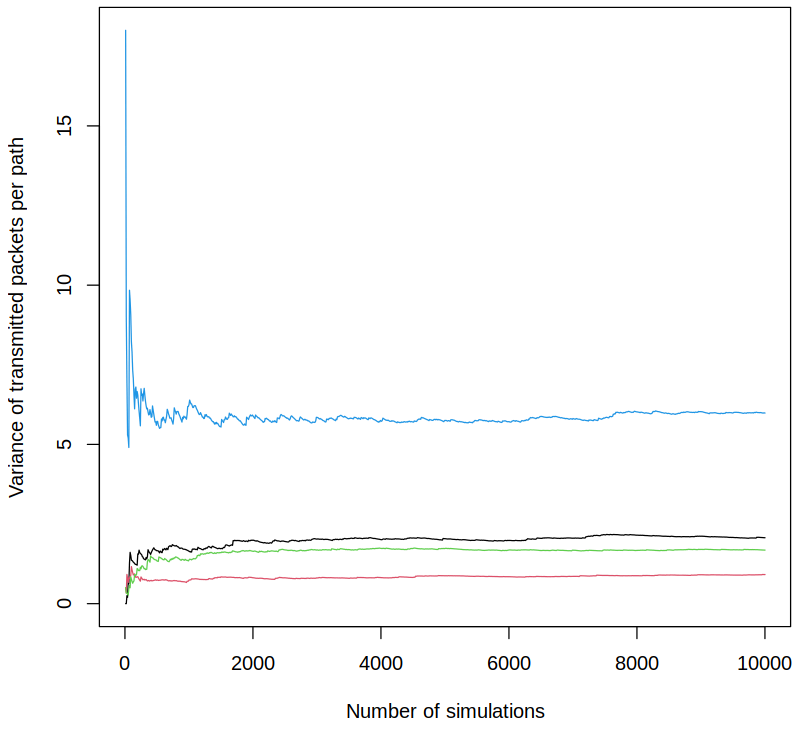
Use the function *Simulation(A,Theta,n.experience)* to simulate 10,50 and 100 times the number of transmitted packets per path. Study the mean and the variance of the number of transmitted packets per path for 10,50 and 100 simulations. What do you observe?

In [73]:
```
n.p <- 4
A <- matrix(c(1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1), nrow = n.p, ncol = 3, byr
n.experience <- c(10, 10, 10, 30)
Theta <- c(0.8, 0.9, 0.99)

max.simulation.nb <- 10001
step <- 10
simulations <- c()
mean.list <- c()
var.list <- c()

for(i in seq(1, max.simulation.nb, by = step)){
    simulations <- c(simulations, Simulation(A, Theta, n.experience))
    # CHATGPT
    # Calculate the mean for each component across all simulations
    mean_vector <- colMeans(matrix(simulations, ncol = n.p, byrow = TRUE))

    # Calculate the variance for each component across all simulations
    var_vector <- apply(matrix(simulations, ncol = n.p, byrow = TRUE), 2, va

    # Append the mean and variance vectors to the respective lists
    mean.list <- rbind(mean.list, mean_vector)
    var.list <- rbind(var.list, var_vector)
}
```

In [74]:
```
# Plot columns in the same plot
matplot(seq(1, max.simulation.nb, by = step), mean.list, type = "l", col = 1
title("Mean of transmitted packets for each path")
```



In [75]:
```
# Plot columns in the same plot
matplot(seq(1, max.simulation.nb, by = step), var.list, type = "l", col = 1:
title("Variance of transmitted packets for each path")
```



The limit number of packets observed in a path is related to the probability of a packet being succesfully transmitted, that is,

$$\mathbb{P}(\text{success}) = \prod_{l=1}^{L_p} \theta_l$$

As can be observed, the mean values of transmitted packets per path stabilize as the number of iterations grows. However, the variance, while bounded, does not tend to 0 as the number of iterations grows to infinity, but actually seems to have a different limit value. This is related to the Cramer Rao bound, which states that any unbiased estimator $\hat{\theta}$ is limited by

$$Cov(\hat{\theta}) \geq I(\theta)^{-1}$$

where $I(\theta)$ is the Fischer information.

Estimation of the success rate using likelihood

3/ The maximum likelihood function computes the maximum likelihood function associated to our statistical problem. Use the invariance principle ML to obtain a simple form of the MLE and complete the function *Maximum.Likelihood*. Then test your function with $Theta.experience = [0.7, 0.8, 0.9]$ and $n.experience = [500, 500, 500, 500]$. Are you able to recover the true value of θ?

In [76]:
```
log.likelihood <- function(A,Theta,x,n) {
    ML <- 0
    for(l in 1:length(n)){
        q_l <- prod(Theta[A[l,]==1])
        ML <- ML + log(choose(n[l], x[l])) + x[l] * log(q_l) + (n[l] - x[l])
    }
    return(ML)
}
```

The maximum likelihood is given by

$$ML(\boldsymbol{\theta}, \boldsymbol{x}) = log\left(\prod_{l=1}^{L} \binom{n_l}{x_l} \left(\prod_{A_{li}=1} \theta_i\right)^{x_l} \left(1 - \prod_{A_{li}=1} \theta_i\right)^{n_l - x_l}\right)$$

A numeric optimisation can then be performed to retrieve the estimator $\hat{\boldsymbol{\theta}}$ by maximising the ML.

In [77]:
```
Theta.experience <- c(0.7, 0.8, 0.9)
n.experience <- c(500, 500, 500, 500)
x <- Simulation(A,Theta.experience, n.experience)
```

```
Maximum.Likelihood <- function(A, x, n) {

    # Maximize the likelihood function
    result <- optim(par = Theta.experience, fn = log.likelihood, A = A, x =

    # Extract the estimated Theta
    Theta_hat <- result$par

    return(Theta_hat)
}

Theta_hat <- Maximum.Likelihood(A, x, n.experience)

cat("Estimated Theta:", Theta_hat)
```

```
Estimated Theta: 0.7043035 0.8085139 0.8866731
```

The estimator of $\boldsymbol{\theta}$ works as expected.

Alternatively, through a derivation of the ML function with respect to each parameter $q_l = \prod_{A_{li}=1} \theta_i$, it can easily be shown that the MLE of q_l is given by

$$q_l = \frac{x_l}{n_l}$$

By solving the equation system, all of the θ_i can be retrieved with a closed formula just from the observation of $\boldsymbol{x}$.

Fisher Information

4/ We have seen during the class that it is possible to compute the Fisher information using the Hessian matrix.

- Please complete the code below by adding the likelihood function of our model;
- Compute the Hessian of the log likelihood using the *Hessian* function. We just recall that the deterministic Hessian matrix is equal to:

$$H(\theta; x) = [[\frac{\partial^2 l(\theta; x)}{\partial \theta_i \theta_j}]]_{1 \leq i,j \leq I}$$

In [78]:
```
Likelihood <- function(A,Theta,x,n) {
    Likelihood.function<-1
    for(p in 1:dim(A)[1]){
        # q_p : Success probability for path p
        q_p <- prod(Theta[A[p,]==1])
        Likelihood.function <- Likelihood.function * choose(n[p], x[p]) * q_
    }
    return(Likelihood.function)
}
```

```
cat("\n Log of Likelihood function:", log(Likelihood(A, Theta.experience, x,
cat("\n Log-likelihood function:", log.likelihood(A, Theta.experience, x, n.
```

```
Log of Likelihood function: -13.75902
Log-likelihood function: -13.75902
```

In [79]:
```
### Add the computation of the Hessian hessian(log.likelihood, Theta.experie
log.likelihood2 <- function(Theta){
    return(log.likelihood(A, Theta, x, n.experience))
}

Hessian_matrix <- hessian(log.likelihood2, Theta.experience)

print(Hessian_matrix)
```

```
          [,1]      [,2]      [,3]
[1,] -4094.413 -2005.532 -2172.364
[2,] -2005.532 -3830.748 -2516.772
[3,] -2172.364 -2516.772 -3307.014
```

5/ We have seen during the class that the Fisher Information is equal to:

$$I(\theta) = E[-H(\theta; X)].$$

A possible approximation is the following. Perform n experiments which provide a realisation $\{1, \ldots, x_n\}$. Then compute:

$$\overline{H}(\theta, n) = \frac{-\sum_{k=1}^{n} H(\theta; x_n)}{n}.$$

It can be proven that

$$\lim_{n \to +\infty} \overline{H}(\theta, n) = I(\theta).$$

The following code is computing $\overline{H}(\theta, n)$. Explain the purpose of each line of the code. Explain the inputs and the outputs of the *Average.fisher.Information* function.

In [80]:
```
Average.fisher.Information<-function(A,Theta.experience,n.experience,nb.iter
    Sample.fisher.information <- array(0, c(length(Theta.experience),length(
    Mean.fisher.information <- array(0, c(length(Theta.experience),length(Th
    for(n in 1:nb.iterations){
         Experience <- Simulation(A,Theta.experience,n.experience)
            # Define log.likelihood function from previously defined likelih
            log.likelihood<-function(Theta){
                return(log(Likelihood(A,Theta,Experience,n.experience)))
                }
        # Compute hessian so as to add new sample
        z<-hessian(log.likelihood, Theta.experience)
        # Add sample matrix to current iteration
        Sample.fisher.information[,,n]<-z
        # Sum value to the mean
        Mean.fisher.information <-z+Mean.fisher.information
```

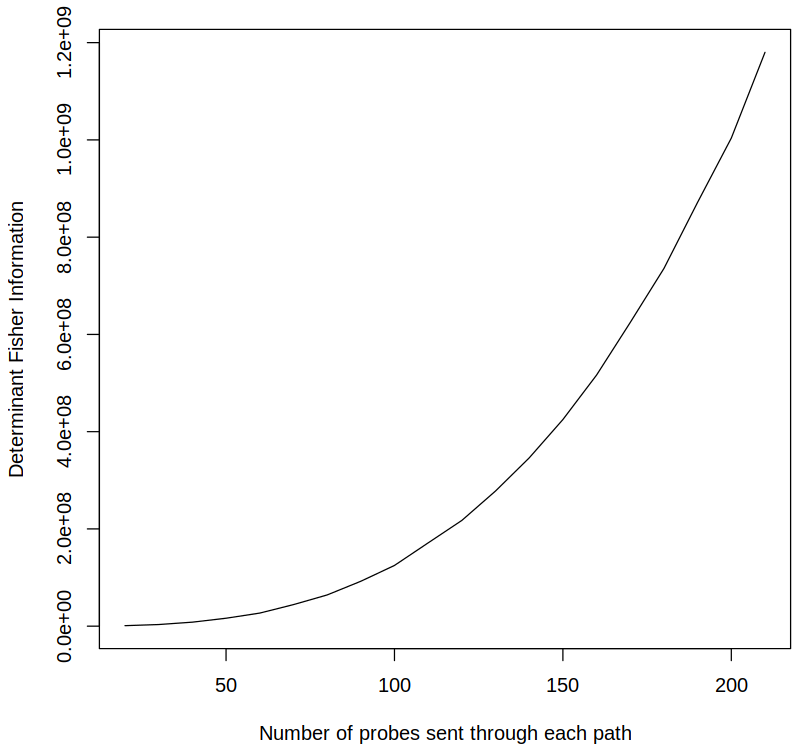
```
    }
    # Divide mean by number of elements and return
    return(list(Sample.fisher.information,Mean.fisher.information/nb.iterati
}
```

6/ Complete the function *det.fisher.information* which is returning the determinant of the average fisher information. Try to increase the number of probes per path from 10 to 200 and observe the evolution of the determinant of the average fisher information. What can you conclude?

In [81]:
```
det.fisher.information<-function(A,Theta.experience,n.experience,nb.iteratic
    fisher_information <- Average.fisher.Information(A,Theta.experience,n.ex
    det_Fisher_information <- det(matrix(unlist(fisher_information[2]), ncol
}
```

In [82]:
```
Theta.experience=c(0.7,0.8,0.9)
n.experience <-c(10,10,10,10)

det.vector<-c()
for(i in 1:20){
    det.vector<-c(det.vector,-det.fisher.information(A,Theta.experience,n.ex
}
```

In [83]:
```
plot(n.experience+10*c(1:20),det.vector,type='l',xlab="Number of probes sent
```



In [84]:
```
-det.fisher.information(A,Theta.experience,c(200, 200, 200, 200),100)
```

1016406950.67853

As the number of probes sent through each path increases, the Determinant of the Fisher information increases as well. This implies that unbiased estimators obtained in a scenario where there are more probes being sent will have lower variance. This is true because the covariance matrix is restricted by the Cramer-Rao bound which is the inverse of the Fisher information matrix.

7/ Design an algorithm that at the same time optimizes the probing strategy and estimates θ. (Think of the EM algorithm)

In [85]:
```
Theta.experience=c(0.7,0.8,0.9)
n.experience <-c(210, 210, 210, 210)
x <- Simulation(A, Theta.experience, n.experience)
Maximum.Likelihood(A, x, n.experience)
```

0.746432571106981 · 0.813674344337756 · 0.886908902970028

In [86]:
```
# Note: This function has some optimisation library problem
# If the above cell is erased, and a run all is done, this cell will weirdly
Estimate.theta <-function(A, Theta.list, nb.iterations, threshold){
    # The idea of this function is to estimate the value of theta while mini
    n.list <- c(200,200,200,200)
    j <- 1
    for(i in 1:nb.iterations){
        x <- Simulation(A, Theta.list, n.list + j*10)
        theta_hat <- Maximum.Likelihood(A, x, n.list + j*10)
        j <- 0
        det <- 0
        # Find number of probes for given threshold
        while(det < threshold){
            j <- j+1
            det <- -det.fisher.information(A, Theta.list,n.list+j*10, 100)
        }
    }
    return(theta_hat)
}

Theta.list <- c(0.7, 0.8, 0.9)
threshold <- 1E9
Estimate.theta(A, Theta.list, 10, threshold)
```

0.730502788784212 · 0.805168954712096 · 0.919253508130685