simulation_MG1_800G

2024-11-29

```r
source("func/igraph_functions.R")
source("func/general_function_base.R")
source("func/simmer_function_base.R")
```

#Input data

The **average packet size** is taken from: Amsterdam Internet Exchange Ethernet Frame Size Distribution, statistics available online at https://stats.ams-ix.net/sflow/size.html, accessed on July 2023

Packet sizes in Bytes: 64-127, 128-511, 512 - 1023, 1024 - 1513, 1514, more than 1515.

With their probabilities: 0.332, 0.054, 0.033, 0.037, 0.346, 0.146, 0.052

```r
PS_size=c((64+127)/2,(128+255)/2,(256+511)/2, (512+1023)/2, (1024+1513)/2, 1514, (1515+9100)/2)
PS_weights=c(33.2/100, 5.4/100, 3.3/100, 3.7/100, 34.6/100, 14.6/100, 5.2/100)
N = sum(PS_size*PS_weights)
N
```

```
## [1] 1019.035
```

```r
var_N <- sum(PS_size^2*PS_weights) - N^2
Cs2 <- var_N/(N^2)

CapacityGbps = 800
#Calculation of delays for different capacities
Load_local = 0.3
Load_regional = 0.5
Load_national = 0.4
```

Input excel file with topology information: links, nodes, traffic

Prefix base - the common name for the nodes in the file.

```r
#topology_choice <- readline(prompt = "Enter 'Tokyo' or 'Milano' to choose the respective topology: ")

# # Define the file paths based on user input
# file_name_v2 <- "input_files/Metro_topology_full_Tokyo.xlsx"
# prefix_base <- "Tokyo_"
# topology_name = "Tokyo"
# print(file_name_v2)

file_name_v2 <- "input_files/Metro_topology_full_Milano.xlsx"
prefix_base <- "Node"
topology_name = "Milano"
print(file_name_v2)
```

```
## [1] "input_files/Metro_topology_full_Milano.xlsx"
```

```
#
# file_name_v2 <- "input_files/Metro_topology_MAN157.xlsx"
# prefix_base <- ""
# topology_name = "MAN157"
# print(file_name_v2)
```

```
nodes_info <- read_excel(file_name_v2, sheet = 1)
links_info <- read_excel(file_name_v2, sheet = 2)
traffic_file <- read_excel(file_name_v2, sheet = 3)
```

```
print(links_info)
```

```
## # A tibble: 202 x 4
##    sourceID destinationID distanceKm capacityGbps
##    <chr>    <chr>              <dbl>        <dbl>
##  1 Node1    Node2                1             0
##  2 Node1    Node3                1             0
##  3 Node1    Node4                1             0
##  4 Node1    Node8                1          3824.
##  5 Node2    Node1                1             0
##  6 Node2    Node5                1          3998.
##  7 Node2    Node6                1             0
##  8 Node2    Node7                1             0
##  9 Node3    Node1                1             0
## 10 Node3    Node4                0.8           0
## # i 192 more rows
```

```
print(nodes_info)
```

```
## # A tibble: 52 x 11
##    node_name node_code `Node Type`  `Central office type` Reference Regional C~1
##    <chr>     <chr>     <chr>        <chr>                 <chr>
##  1 Node1     HL2       Metro Core   Regional CO           Node1
##  2 Node2     HL2       Metro Core   Regional CO           Node2
##  3 Node3     HL2       Metro Core   Regional CO           Node3
##  4 Node4     HL2       Metro Core   Regional CO           Node4
##  5 Node5     HL2       Metro Core   Regional CO           Node5
##  6 Node6     HL2       Metro Core   Regional CO           Node6
##  7 Node7     HL2       Metro Core   Regional CO           Node7
##  8 Node8     HL2       Metro Core   Regional CO           Node8
##  9 Node9     HL3       Metro Core ~ National CO           Node8
## 10 Node10    HL3       Metro Core ~ National CO           Node8
## # i 42 more rows
## # i abbreviated name: 1: `Reference Regional CO`
## # i 6 more variables: `Reference National CO` <chr>, Households <dbl>,
## #   `Macro cells sites` <dbl>, `Small cell sites` <dbl>,
## #   `Twin Regional CO` <chr>, `Twin National CO` <chr>
```

```r
print(traffic_file)
```

```
## # A tibble: 104 x 4
##    sourceID destinationID trafficGbps service
##    <chr>    <chr>               <dbl> <chr>
##  1 Node1    Node9               1945. CWB
##  2 Node2    Node16              1671. CWB
##  3 Node3    Node11              1498. CWB
##  4 Node4    Node13              3015. CWB
##  5 Node5    Node16              1805. CWB
##  6 Node6    Node19              3565. CWB
##  7 Node7    Node9               2618. CWB
##  8 Node8    Node9               1916. CWB
##  9 Node9    Node9               3798. CWB
## 10 Node10   Node10              3911. CWB
## # i 94 more rows
```

Definition of national and regional Cental Offices (COs).

```r
national_nodes <- c()
regional_nodes <- c()

for (i in seq_along(nodes_info$node_code)) {
  if (nodes_info$node_code[i] == "HL2") {
    national_nodes <- c(national_nodes, i)
  }
  if (nodes_info$node_code[i] == "HL3") {
    regional_nodes <- c(regional_nodes, i)
  }
}

cat("National nodes:", national_nodes, "\n")
```

```
## National nodes: 1 2 3 4 5 6 7 8 21 22 23 24
```

```r
cat("Regional nodes:", regional_nodes, "\n")
```

```
## Regional nodes: 9 10 11 12 13 14 15 16 17 18 19 20 40 41 42 43 44
```

Functions

Vysochanskij–Petunin's bound calculation function

Inputs: - **delay__hops**: Vector of delay at each hop - **a**: Upper bound percentile from 0 to 1

Output: - Vysochanskij–Petunin's upper bound

```r
func_bounds_VP <- function(delay_hops, a)
{
  mu = sum(delay_hops)
```

```
  sigma = sqrt(sum(delay_hops^2))
  k <- sqrt(4/9/(1-a))
  Prop_VP <- 1 - 4/9/(k^2)
  upper_bound_VP <- k * sigma + mu
  return(upper_bound_VP)
}
```

##Simmer simulation

Igraph calculations

Building the graph:

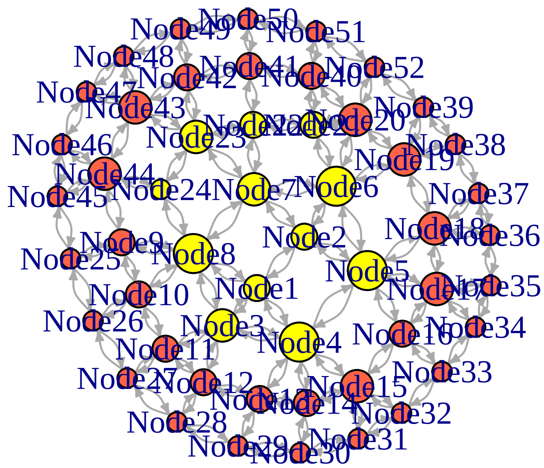
```
g <- graph_from_data_frame(links_info, directed = TRUE, vertices = nodes_info)
```

Calculations of the capacity in p/s

```
E(g)$Distance <- E(g)$distanceKm
E(g)$Definition <- paste0(as_edgelist(g)[,1],"->",as_edgelist(g)[,2])
E(g)$Capacity <- E(g)$capacityGbps*10^9/(8*N)
```

Plot graph

```
#plot graph
V(g)$color <- "tomato"
V(g)$color[national_nodes] <- "yellow"

deg <- degree(g, mode="all")
V(g)$size <- deg*1.5
l <- layout_nicely(g)
plot(g, edge.arrow.size=.3, vertex.label = V(g)$name, edge.curved=.5, layout=l)
```



#Igraph calculations ## Total traffic calculatons ## Calculations of the load, average number of packets, queueing and propagation delays: Load for Local COs 0.3; Regional COs 0.5; Rational COs 0.4

```
c(g_c10, data_av, data_99) := simulation_igraph(nodes_info, links_info, traffic_file, Capacity = 10, cal
```

#Simmer simulation Including queuing, transmission, propagation delay Output table with links information

Simulation of all traffic flows, comparision of experimental with theretical

#Results

```
links_info_df
```

```
## # A tibble: 202 x 5
##    sourceID destinationID distanceKm capacityGbps prop_delay_s
##    <chr>    <chr>              <dbl>        <dbl>        <dbl>
##  1 Node1    Node2                1            0       0.000005
##  2 Node1    Node3                1            0       0.000005
##  3 Node1    Node4                1            0       0.000005
##  4 Node1    Node8                1         3824.      0.000005
##  5 Node2    Node1                1            0       0.000005
##  6 Node2    Node5                1         3998.      0.000005
##  7 Node2    Node6                1            0       0.000005
##  8 Node2    Node7                1            0       0.000005
##  9 Node3    Node1                1            0       0.000005
## 10 Node3    Node4                0.8          0       0.000004
## # i 192 more rows
```

```
traffic_file_v2
```

```
## # A tibble: 72 x 11
##    sourceID destinationID trafficGbps service traffic_ps latencyPropTransQueui~1
##    <chr>    <chr>               <dbl> <chr>        <dbl>                   <dbl>
##  1 Node1    Node9                 240 CWB      29439617.              0.00000744
##  2 Node2    Node16                240 CWB      29439617.              0.00000744
##  3 Node3    Node11                240 CWB      29439617.              0.00000337
##  4 Node4    Node13                240 CWB      29439617.              0.00000337
##  5 Node5    Node16                240 CWB      29439617.              0.00000337
##  6 Node6    Node19                240 CWB      29439617.              0.00000337
##  7 Node7    Node9                 240 CWB      29439617.              0.00000694
##  8 Node8    Node9                 240 CWB      29439617.              0.00000337
##  9 Node20   Node19                240 CWB      29439617.              0.00000337
## 10 Node21   Node40                240 CWB      29439617.              0.00000337
## # i 62 more rows
## # i abbreviated name: 1: latencyPropTransQueuing_theor_s_mg1
## # i 5 more variables: latencyPercentile99_theor_s_mg1 <dbl>, Delay_sim_s <dbl>,
## #   Delay_th_s <dbl>, D99_sim_s <dbl>, VPbound_99th_s <dbl>
```

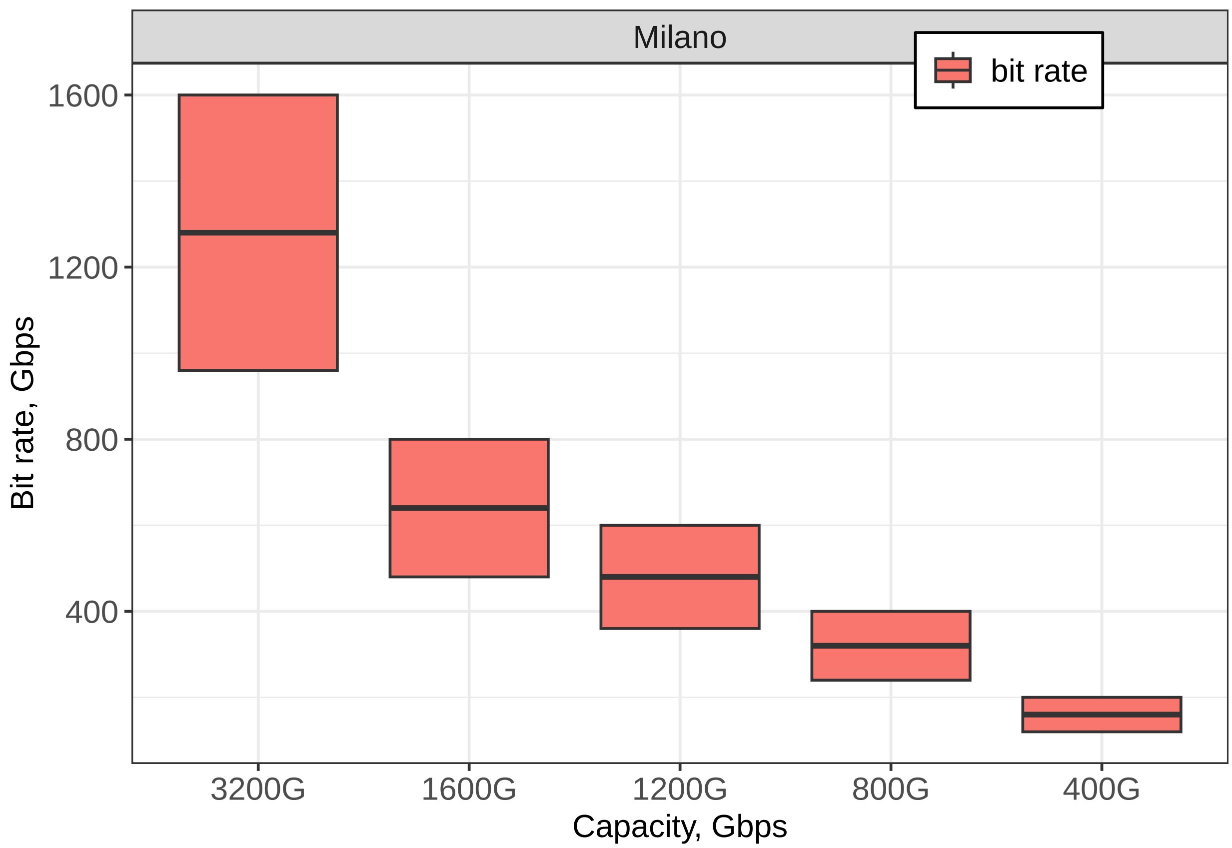
#Calculation of delays for different capacities

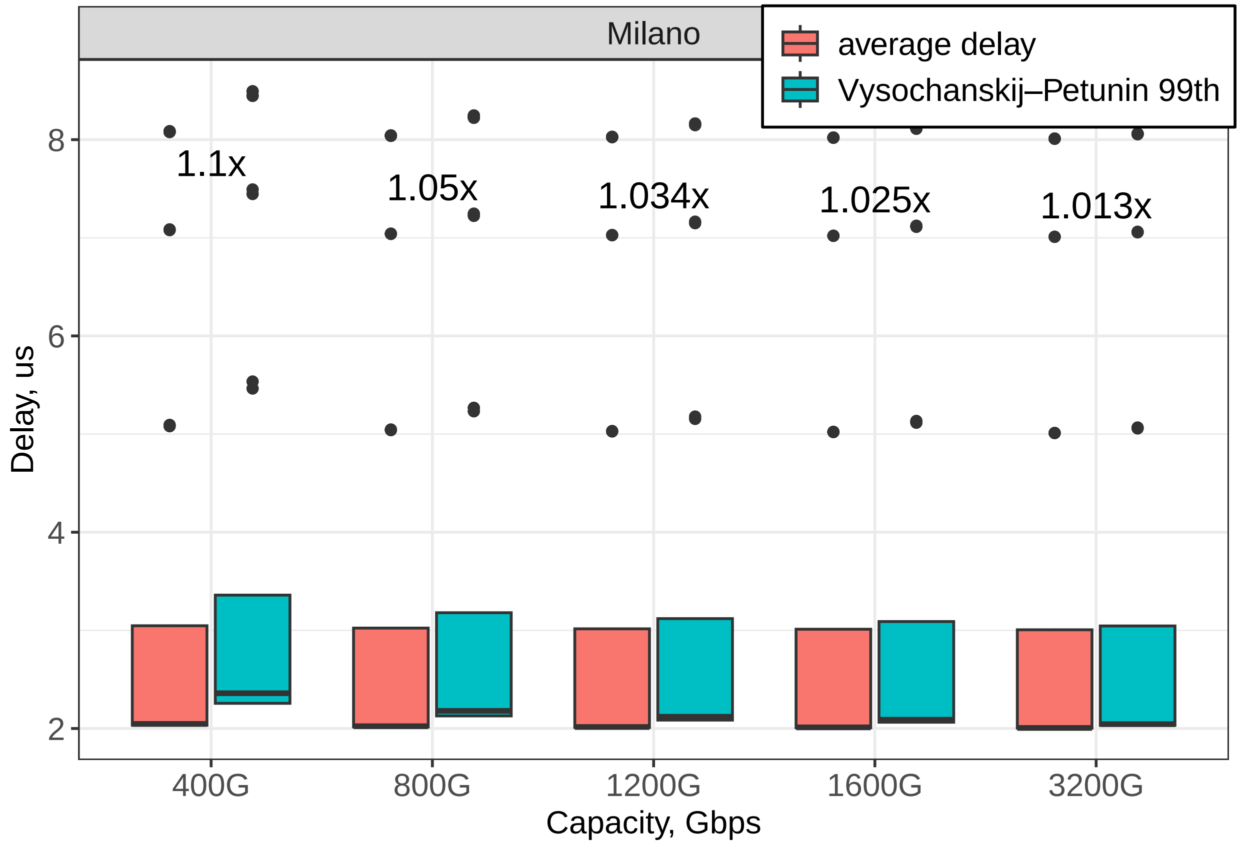
```
capacities = c(400, 800, 1200, 1600, 3200)
c(g_c400_dg_l05, data_av_c400_dg_l05, data_99_c400_dg_l05, trafficGbps_c400_dg_l05) :=simulation_igraph
c(g_c800_dg_l05, data_av_c800_dg_l05, data_99_c800_dg_l05, trafficGbps_c800_dg_l05) :=simulation_igraph
c(g_c12_dg_l05, data_av_c12_dg_l05, data_99_c12_dg_l05, trafficGbps_c12_dg_l05) :=simulation_igraph(node
c(g_c16_dg_l05, data_av_c16_dg_l05, data_99_c16_dg_l05, trafficGbps_c16_dg_l05) :=simulation_igraph(node

c(g_c32_dg_l05, data_av_c32_dg_l05, data_99_c32_dg_l05, trafficGbps_c32_dg_l05) :=simulation_igraph(node
```

```
text_size = 12

# Boxplot for trafficGbps (bit rate in Gbps)
df_plot_traffic <- data.frame(capacity = c(
                                            rep("400G", length(trafficGbps_c400_dg_l05)),
                                            rep("800G", length(trafficGbps_c800_dg_l05)),
                                            rep("1200G", length(trafficGbps_c12_dg_l05)),
                                            rep("1600G", length(trafficGbps_c16_dg_l05)),
                                            rep("3200G", length(trafficGbps_c32_dg_l05))),
                              value = c(trafficGbps_c400_dg_l05,
                                        trafficGbps_c800_dg_l05,
                                        trafficGbps_c12_dg_l05,
                                        trafficGbps_c16_dg_l05,
                                        trafficGbps_c32_dg_l05),
                              type_of_calc = "bit rate",
                              city = c(rep(topology_name,length(trafficGbps_c32_dg_l05)*10)))

ggplot(df_plot_traffic, aes(x = fct_reorder(capacity, value, .desc = TRUE), y = value, fill = type_of_ca
  geom_boxplot() +
  theme_bw() +
  theme(
    legend.title = element_blank(),
    legend.position = c(0.8, 0.99),
```

```
  axis.title.y = element_text(size = text_size),
  axis.title.x = element_text(size = text_size),
  legend.text = element_text(size = text_size),
  legend.background = element_rect(colour = "black"),
  axis.text = element_text(size = text_size),
  strip.text = element_text(size = text_size)
) +
facet_grid(. ~ city ) +
ylab("Bit rate, Gbps") +
xlab("Capacity, Gbps")
```

```
## Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
## 3.5.0.
## i Please use the `legend.position.inside` argument of `theme()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



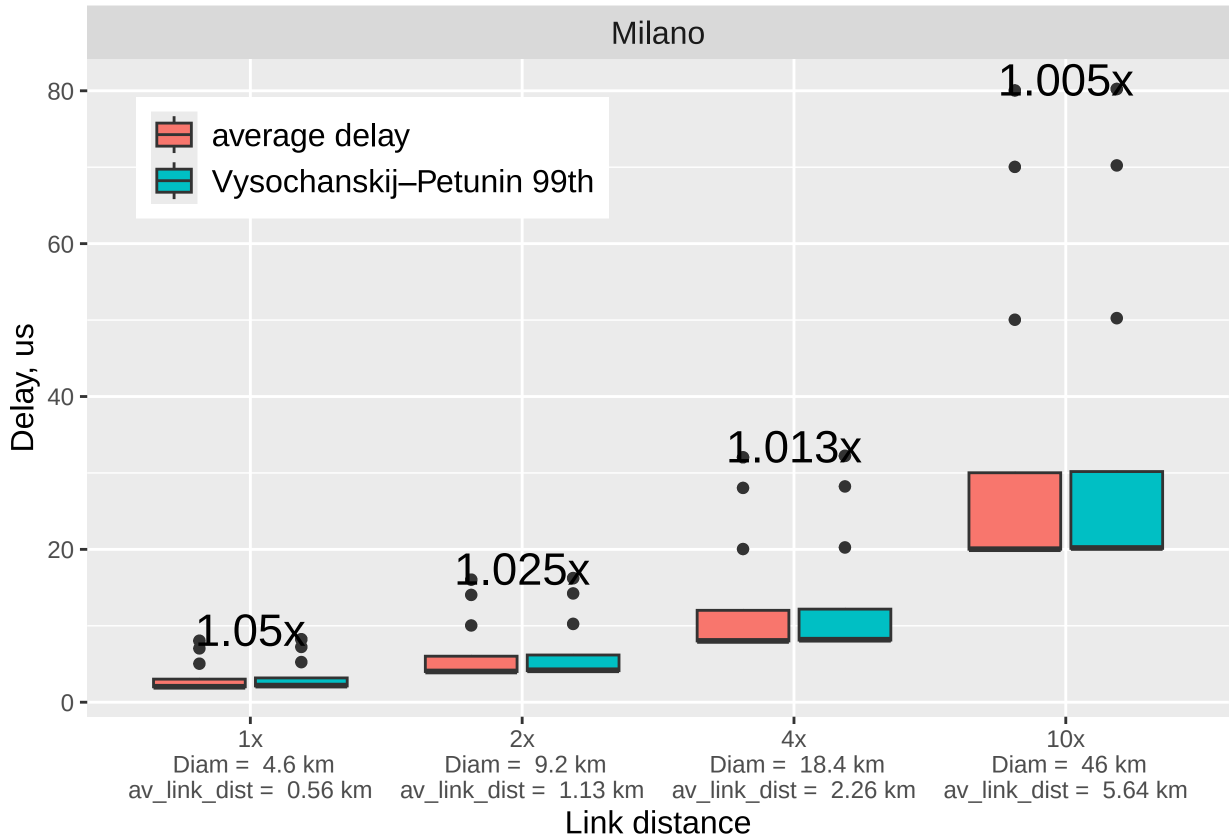


As shown in the plot, the queuing delay does not significantly affect the e2e delay. This is because it is too low compared to the propagation delay and does not result in noticeable changes for changing capacity higher than 800G.

#Calculation of delays for different distances

```
#Calculation of delays for different distances
c(g_c100_d05g_l05, data_av_c100_d05g_l05, data_99_c100_d05g_l05) :=simulation_igraph(nodes_info, links_i

#Calculation of delays for different distances
c(g_c100_dg_l05, data_av_c100_dg_l05, data_99_c100_dg_l05) :=simulation_igraph(nodes_info, links_info, 

c(g_c100_d2g_l05, data_av_c100_d2g_l05, data_99_c100_d2g_l05) :=simulation_igraph(nodes_info, links_inf
c(g_c100_d4g_l05, data_av_c100_d4g_l05, data_99_c100_d4g_l05) :=simulation_igraph(nodes_info, links_inf
c(g_c100_d10g_l05, data_av_c100_d10g_l05, data_99_c100_d10g_l05) :=simulation_igraph(nodes_info, links_i
```

```
## [1] "0.5x diam 2.3 km av link dist 0.28 km"
```



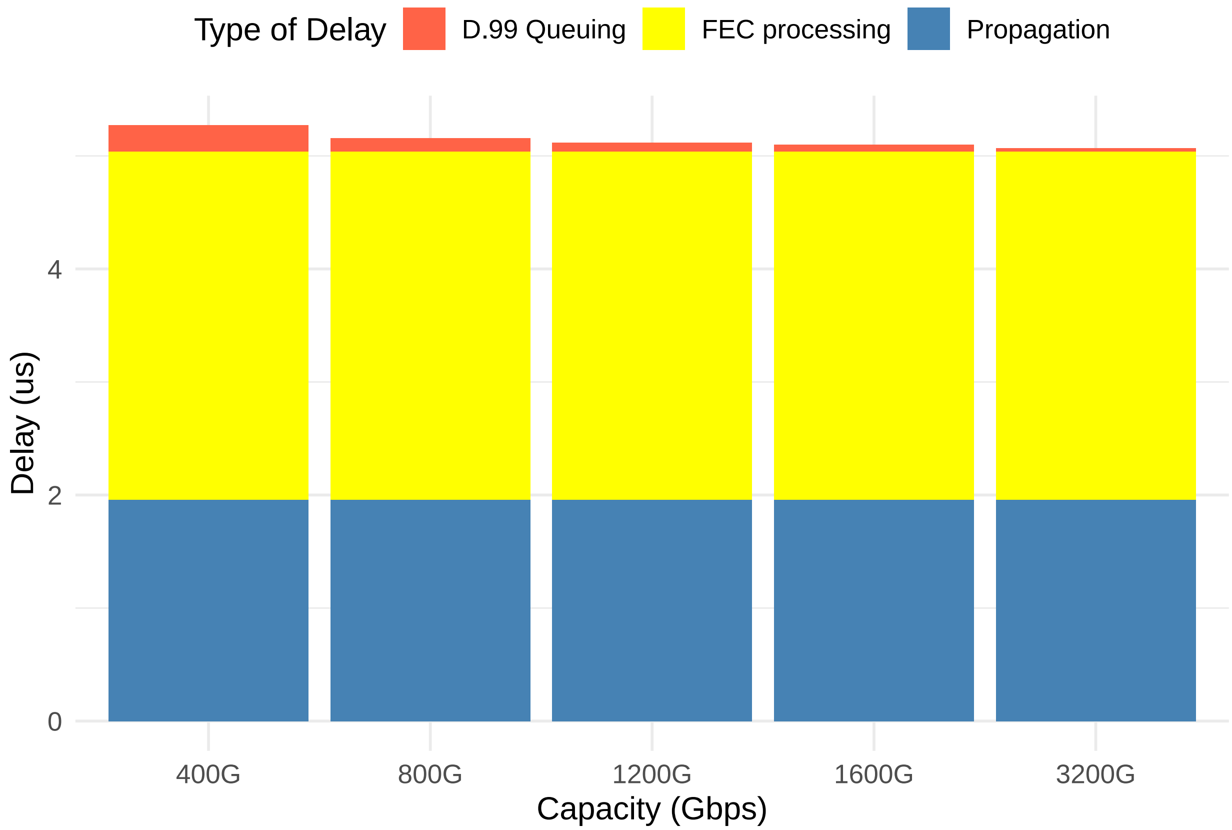

```
prop_queue_tr_delay_e2e <- function(g, traffic_file, Cs2){

  E(g)$Queue_Delay <- ifelse(E(g)$Traffic == 0, 0, E(g)$Load/(1-E(g)$Load)/E(g)$Traffic*(Cs2 + 1)/2) #M
  latencyPropTransQueuing_theor_s <- rep(0, nrow(traffic_file))
  latencyPercentile99_theor_s <- rep(0, nrow(traffic_file))

  latencyProp_s <- rep(0, nrow(traffic_file))
  latencyPercentile99_theor_TransQueuing_s <- rep(0, nrow(traffic_file))
  processing_FEC_s <- rep(0, nrow(traffic_file))

  for (line in 1:nrow(traffic_file))
  {
    if(traffic_file$traffic_ps[line] != 0){
      vertex_sourse <- which(V(g)$name==traffic_file$sourceID[line])
      vertex_destination<- which(V(g)$name==traffic_file$destinationID[line])
      path <- shortest_paths(g, vertex_sourse, vertex_destination,
                             weights = NULL,
                             output = "both",
                             algorithm = c("automatic"))
      latencyPropTransQueuing_theor_s[line] = sum(E(g)[path[["epath"]][[1]]]$Queue_Delay) + sum(E(g)[pa
      latencyPercentile99_theor_s[line] <- func_bounds_VP(E(g)[path[["epath"]][[1]]]$Queue_Delay, 0.99)

      latencyProp_s[line] = sum(E(g)[path[["epath"]][[1]]]$Prop_Delay)
      latencyPercentile99_theor_TransQueuing_s[line] <- func_bounds_VP(E(g)[path[["epath"]][[1]]]$Queue_
      processing_FEC_s[line] <- 4*1e-6*length(path[["epath"]][[1]])
    }
```

```r
    else {
      latencyPropTransQueuing_theor_s[line] <- 0
      latencyPercentile99_theor_s[line] <- 0
      latencyProp_s[line] <- 0
      latencyPercentile99_theor_TransQueuing_s[line] <- 0
      processing_FEC_s[line] <- 0

    }
  }

  return(list(g, latencyPropTransQueuing_theor_s, latencyPercentile99_theor_s, latencyProp_s, latencyPe
}


simulation_igraph <- function(nodes_info, links_info, traffic_file, Capacity = 10, calc_dist =  FALSE, 

  traffic_file$traffic_ps <- traffic_file$trafficGbps*10^9/(8*N)
  g = graph_from_data_frame(links_info, directed = TRUE)
  E(g)$capacityGbps <- Capacity
  for (NCO in national_nodes){
    a <- (filter(traffic_file, traffic_file$destinationID == V(g)$name[NCO]))
    size <- length(a$destinationID)
    E(g)$capacityGbps[incident(g, NCO, mode = c("in"))] <- Capacity*size
  }
  E(g)$Capacity <- E(g)$capacityGbps*1e9/(8*N) #p/s
  ############distance put original distance(calc_dist = 1) or custom distance (calc_dist = 1) with par
  if( calc_dist == TRUE ) {
    E(g)$Distance = E(g)$distanceKm * distance
  }
  else {
    E(g)$Distance <- E(g)$distanceKm
  }
  ####Load of the links
  #for Local CO
  E(g)$Load <- Load_local
  #national COs
  for (NCO in national_nodes){
    E(g)$Load[incident(g, NCO, mode = c("in"))] <- Load_national
  }
  #regional COs
  for (RCO in regional_nodes){
    E(g)$Load[incident(g, RCO, mode = c("in"))] <- Load_regional
  }

  E(g)$Traffic <- E(g)$Capacity * E(g)$Load
  E(g)$Ni = E(g)$Load/(1-E(g)$Load) # average number of packets in each system
  E(g)$Prop_Delay <- 5*10^(-6)*E(g)$Distance
  c(g, traffic_file$latencyPropTransQueuing_theor_s_mg1, traffic_file$latencyPercentile99_theor_s_mg1, 

  data_prop <- latencyProp_s_mg1
  data_99_TransQueuing <- latencyPercentile99_theor_TransQueuing_s_mg1
```

```r
  assign("traffic_file", traffic_file, .GlobalEnv)

  links_info$prop_delay_s <- 5*10^(-6)*links_info$distanceKm
  assign("links_info_df", links_info, .GlobalEnv)

  #removing the local traffic
  data_av <- data_av[data_av != 0]
  data_99 <- data_99[data_99 != 0]
  trafficGbps <- E(g)$Capacity * E(g)$Load[E(g)$Capacity * E(g)$Load != 0]/1e9*(8*N) #Gbps

  return(list(g, data_prop, data_99_TransQueuing, trafficGbps, processing_FEC_s))


}
```

```r
capacities = c(400, 800, 1200, 1600, 3200)
c(g_c400_dg_l05, data_prop_c400_dg_l05, data_99_TransQueuing_c400_dg_l05, trafficGbps_c400_dg_l05, proce
c(g_c800_dg_l05, data_prop_c800_dg_l05, data_99_TransQueuing_c800_dg_l05, trafficGbps_c800_dg_l05, proce
c(g_c12_dg_l05, data_prop_c12_dg_l05, data_99_TransQueuing_c12_dg_l05, trafficGbps_c12_dg_l05, processin
c(g_c16_dg_l05, data_prop_c16_dg_l05, data_99_TransQueuing_c16_dg_l05, trafficGbps_c16_dg_l05, processin

c(g_c32_dg_l05, data_prop_c32_dg_l05, data_99_TransQueuing_c32_dg_l05, trafficGbps_c32_dg_l05, processin
```

```r
# Create a data frame with mean propagation and queuing delays for each capacity
df_plot_delay <- data.frame(
  capacity = factor(c("400G", "800G", "1200G", "1600G", "3200G"),
                    levels = c("400G", "800G", "1200G", "1600G", "3200G")),
  delay_type = rep(c("Propagation", "D.99 Queuing", "FEC processing"), each = 5),
  delay_value = c(
    mean(data_prop_c400_dg_l05), mean(data_prop_c800_dg_l05), mean(data_prop_c12_dg_l05), mean(data_prop
    mean(data_99_TransQueuing_c400_dg_l05), mean(data_99_TransQueuing_c800_dg_l05), mean(data_99_TransQu
    mean(processing_FEC_s_c400_dg_l05), mean(processing_FEC_s_c800_dg_l05), mean(processing_FEC_s_c12_dg

  )
)

# Plot the barplot
ggplot(df_plot_delay, aes(x = capacity, y = delay_value * 1e6, fill = delay_type)) +
  geom_bar(stat = "identity", position = "stack") +
  theme_minimal() +
  labs(
    x = "Capacity (Gbps)",
    y = "Delay (us)",
    fill = "Type of Delay"
  ) +
  scale_fill_manual(values = c("Propagation" = "steelblue", "D.99 Queuing" = "tomato", "FEC processing"
  theme(
    text = element_text(size = 12),
    legend.position = "top"
  )
```




```
df_Milano = data.frame(prop = c(mean(data_prop_c400_dg_l05), mean(data_prop_c800_dg_l05), mean(data_prop
                       queuing = c( mean(data_99_TransQueuing_c400_dg_l05), mean(data_99_TransQueuing_c8
```

#Creating the report file

```
update_file(links_info_df, traffic_file_v2, file_name = paste0(topology_name, "_topology_result.xlsx"),
```