Queuing_envelope_mg1

2024-04-24

Approximating Queuing Delay from M/GI/1 to M/M/1 Envelope

FUNCTIONS

Function for Calculating of M/M/1 envelope load for M/G/1

```
envelope_load_calc <- function(Capacity_Gbps, k_num, Load, PS_size, PS_weights)
{
  mg1_packets <- simmer_mg1(Capacity_Gbps, Load, PS_size, PS_weights, k_num = 1e4)
  cat("Load real M/G/1:", Load, "\n")

  # Step 1: Determine the average delay (E(T))
  E_T_real <- mean(mg1_packets)
  N = sum(PS_size*PS_weights)
  N
  var_N <- sum(PS_size^2*PS_weights) - N^2
  Cs2 <- var_N/(N^2)
  nodes_capacity_Bps = Capacity_Gbps*1e9
  Capacity_ps = nodes_capacity_Bps/(8*N)
  E_X = 1/Capacity_ps

  percentiles_seq <- seq(0.5, 0.99, 0.01)

  n = 1
  df_real <- sapply(percentiles_seq, function(x) quantile(mg1_packets, x))*1e6 #real
  rho_env = seq(from=0.01,to=0.99,by=0.01)
  for (rho in rho_env){
    df_env <- qexp(percentiles_seq,rate = ((1-rho)/E_X))*1e6 # mu s
    if (all(df_real < df_env)){
      cat("Envelope upper bound \n")
      cat("Load Envelope:", rho, "\n")
      cat("E(T) exponential envelope M/M/1:", E_X/(1-rho), "s \n")
      cat("E(T) Real M/GI/1:", E_T_real, "s \n")
      cat("\n")
      break
    }
    n = n + 1
  }
  return(rho)
}
```

INPUT DATA

```r
Capacity_Gbps = 10
k_num = 1e6 #number of simulation packages
Load <- 0.7  # Load
# # #V1
# PS_size <- c(40, 576, 1500)  # Packet sizes in bytes
# PS_weights <- c(7, 4, 1) / 12  # Packet weights
```

Part 1: simulation M/G/1 and envelope M/M/1 calculations

SIMULATION

```r
mg1_packets <- simmer_mg1(Capacity_Gbps, Load, PS_size, PS_weights, k_num)
```

M/G/1 calculations of parameters

```r
real_cdf <- ecdf(mg1_packets)
cat("Load real M/G/1:", Load, "\n")
```

```
## Load real M/G/1: 0.7
```

```r
# Step 1: Determine the average delay (E(T))
E_T_real <- mean(mg1_packets)
N = sum(PS_size*PS_weights)
N
```

```
## [1] 1750.41
```

```r
var_N <- sum(PS_size^2*PS_weights) - N^2
Cs2 <- var_N/(N^2)
nodes_capacity_Bps = Capacity_Gbps*1e9
Capacity_ps = nodes_capacity_Bps/(8*N)
E_X = 1/Capacity_ps
E_T_real_theor <- E_X*Load/(1-Load)*(1+Cs2)/2
D_T_real_theor_90 <- (E_X*Load/(1-Load)*(1+Cs2)/2 + E_X)*log(1/(1-0.90))
D_T_real_theor_99 <- (E_X*Load/(1-Load)*(1+Cs2)/2 + E_X)*log(1/(1-0.99))

#sqrt(var_N)
```

Envelope M/M/1 fitting process

Envelope E(T) M/M/1 should be above real M/G/1 for percentiles from 50% - 99%

```
percentiles_seq <- seq(0.5, 0.99, 0.01)

n = 1
df_real <- sapply(percentiles_seq, function(x) quantile(mg1_packets, x))*1e6 #real
rho_env = seq(from=0.01,to=0.99,by=0.01)
for (rho in rho_env){
  df_env <- qexp(percentiles_seq,rate = ((1-rho)/E_X))*1e6 # mu s
  if (all(df_real < df_env)){
    cat("Envelope upper bound \n")
    cat("Load Envelope:", rho, "\n")
    cat("E(T) exponential envelope M/M/1:", E_X/(1-rho), "s \n")
    cat("E(T) Real M/GI/1:", E_T_real, "s \n")
    break
  }
  n = n + 1
}
```

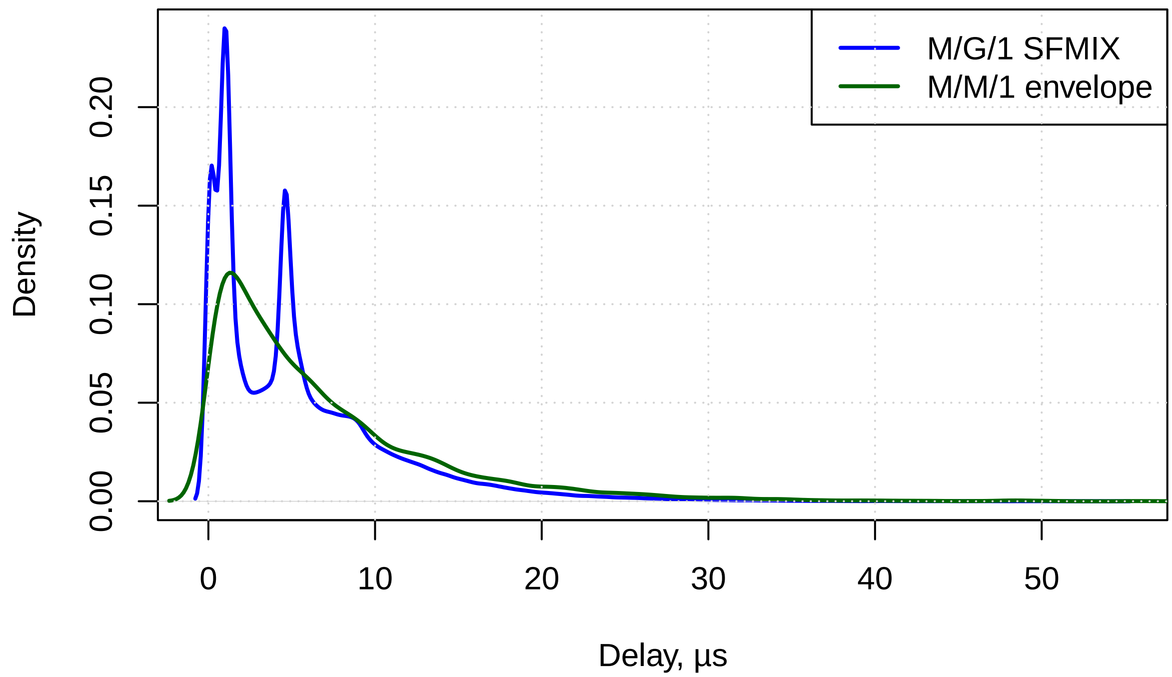
```
## Envelope upper bound
## Load Envelope: 0.79
## E(T) exponential envelope M/M/1: 6.668229e-06 s
## E(T) Real M/GI/1: 5.344584e-06 s
```

```
rho_env = rho
envelope_rate = (1-rho)/E_X
```

Plots

PDF of real and envelope

```
plot(density(mg1_packets*1e6), col = "blue", main = "", xlab = "Delay,  s", lwd = 2)
lines(density(rexp(1e4, rate = envelope_rate)*1e6), col = "darkgreen", lwd = 2)
legend("topright", legend = c("M/G/1 SFMIX", "M/M/1 envelope"), col = c("blue", "darkgreen"), lwd = 2)
grid()
```



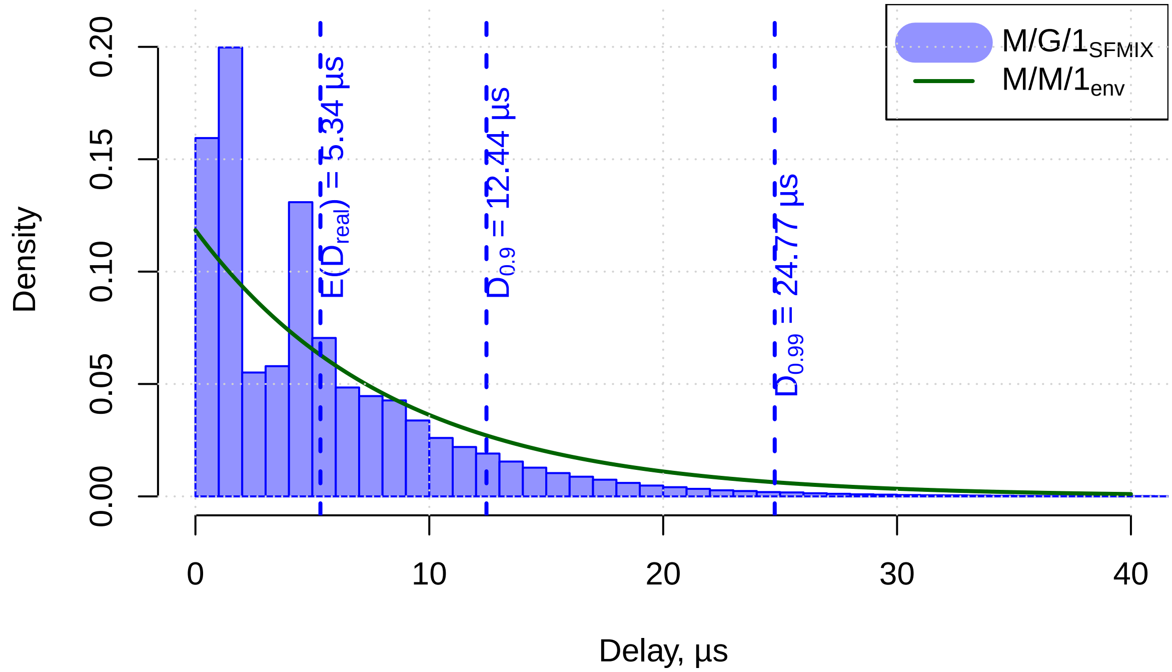

```r
# # Assuming mg1_packets and envelope_rate are already defined
#
# # Plot the histogram of mg1_packets with density option
# hist(mg1_packets * 1e6, breaks = 50, probability = TRUE, col = "#9393ff", main = "", xlab = "Delay,  s
#
# # Add the density line for mg1_packets
# #lines(density(mg1_packets * 1e6), col = "blue", lwd = 2)
#
# # Add the density line for the exponential distribution (envelope)
# dens_env <- density(rexp(1e4, rate = envelope_rate) * 1e6, bw = 0.25)
# dens_env$x[dens_env$x<0] <- 0
# lines(dens_env, col = "darkgreen", lwd = 2)
#
# # Add the legend
# legend("topright", legend = c("M/G/1 SFMIX", "M/M/1 envelope"), col = c("#9393ff", "darkgreen"), lwd
#
# # Add the grid
# grid()
```

```r
lambda <- rho * 1/E_X
# Theoretical M/M/1 delay density function
mm1_delay_density <- function(d, lambda, mu) {
  rho <- lambda / mu
  lambda * (1 - rho) * exp(-lambda * (1 - rho) * d)
}
```

```r
# Define the range of delay values for plotting
delay_values <- seq(0, 50e-6, length.out = 1000)

# Compute the theoretical density values
density_values <- mm1_delay_density(delay_values, lambda, 1/E_X)
```

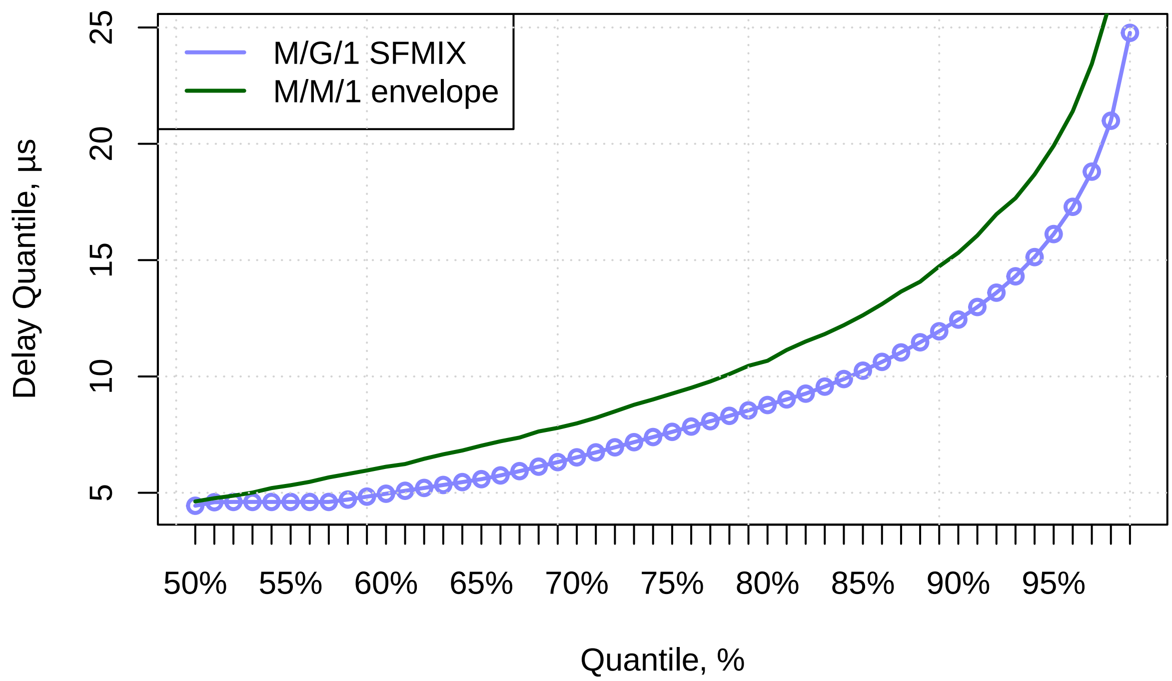
```r
# Assuming mg1_packets, E_X, and rho are defined

# Compute lambda based on rho and E_X
lambda <- rho * 1/E_X

# Theoretical M/M/1 delay density function
mm1_delay_density <- function(d, lambda, mu) {
  rho <- lambda / mu
  lambda * (1 - rho) * exp(-lambda * (1 - rho) * d)
}

xlimit =  40e-6 #s
# Define the range of delay values for plotting
delay_values <- seq(0, xlimit, length.out = 1000)

# Compute the theoretical density values
density_values <- mm1_delay_density(delay_values, lambda, 1/E_X)

# Plot the histogram of mg1_packets with density option
hist(mg1_packets * 1e6, breaks = 50, probability = TRUE, col = "#9393ff", main = "", xlab = "Delay,  s",

lines(delay_values * 1e6, density_values / 1e6, type = "l", col = "darkgreen", lwd = 2,
      xlab = "Delay ( s)", ylab = "Density", main = "Theoretical M/M/1 Delay Distribution")

# Add vertical lines for mean, 0.90 quantile, and 0.99 quantile of mg1_packets
mean_delay <- mean(mg1_packets) * 1e6
quantile_90 <- quantile(mg1_packets, p = 0.9) * 1e6
quantile_99 <- quantile(mg1_packets, p = 0.99) * 1e6

abline(v = mean_delay, col = "blue", lty = 2, lwd = 2)
abline(v = quantile_90, col = "blue", lty = 2, lwd = 2)
abline(v = quantile_99, col = "blue", lty = 2, lwd = 2)

mean_delay = round(mean_delay, 2)
quantile_90 = round(quantile_90, 2)
quantile_99 = round(quantile_99, 2)


# Add labels for the vertical lines with rotated text
text(mean_delay, par("usr")[4] * 0.4, expression(paste("E(D"[real], ") = ", 5.34, "  s")), col = "blue",
text(quantile_90, par("usr")[4] * 0.4, expression(paste("D"[0.90, real], " = ", 12.44 , "  s")), col = "
text(quantile_99, par("usr")[4] * 0.2, expression(paste("D"[0.99, real], " = ", 24.77 , "  s")), col = "

# Add the legend
legend("topright", legend = c(expression("M/G/1"[SFMIX]), expression("M/M/1"[env])), col = c("#9393ff",

# Add the grid
```

```r
grid()
```



Plot the percentiles of real and envelope

```r
df_real <- sapply(percentiles_seq, function(x) quantile(mg1_packets*1e6, x))
df_env <- sapply(percentiles_seq, function(x) quantile(rexp(1e5, rate = envelope_rate), x)*1e6)

# Plot with no additional x-axis labels
plot(df_real, col = "#8585ff", xlab = "Quantile, %", ylab = "Delay Quantile,  s", main = "", xaxt = "n",
lines(df_real, col = "#8585ff", lwd = 2)
lines(df_env, col = "darkgreen", lwd = 2)

# Add x-axis labels for percentiles_seq
axis(1, at = seq_along(percentiles_seq), labels = paste0(percentiles_seq * 100, "%"))
legend("topleft", legend = c("M/G/1 SFMIX", "M/M/1 envelope"), col = c("#8585ff", "darkgreen"), lwd = 2
grid()
```



Plot the log CCDF

```
t_min = min(mg1_packets)
t_max = 2*E_T_real
# Compute the empirical CDF
ecdf_values <- ecdf(mg1_packets)
# Generate x values for the plot E(T)
x_values <- seq(t_min, t_max, length.out = 1e4)
# Compute the empirical CDF values
ecdf_y_values_sim <- ecdf_values(x_values)


envelope_ecdf_val <- envelope_cdf(envelope_rate)
ecdf_y_values_env_upper_bound <- envelope_ecdf_val(x_values)


lwd = 4
plot(x_values, log10(1 - ecdf_y_values_sim), type = "l", xlab = "Delay, s", ylab = "log CCDF", main = "
lines(x_values, log10(1 - ecdf_y_values_env_upper_bound), col = "darkgreen", lty = 2, lwd  = lwd )
legend("topright", legend = c(paste("real CCDF   = ", Load),
                             paste("Envelope CCDF   =", round(rho_env, 2))),
      col = c("blue", "darkgreen" ), lty = c(1, 2), lwd = 2)
```

```
## Warning in (function (s, units = "user", cex = NULL, font = NULL, vfont = NULL,
## : conversion failure on 'real CCDF   = 0.7' in 'mbcsToSbcs': dot substituted
## for <cf>
```

```
## Warning in (function (s, units = "user", cex = NULL, font = NULL, vfont = NULL,
## : conversion failure on 'real CCDF  = 0.7' in 'mbcsToSbcs': dot substituted
## for <81>

## Warning in (function (s, units = "user", cex = NULL, font = NULL, vfont = NULL,
## : conversion failure on 'Envelope CCDF  = 0.79' in 'mbcsToSbcs': dot
## substituted for <cf>

## Warning in (function (s, units = "user", cex = NULL, font = NULL, vfont = NULL,
## : conversion failure on 'Envelope CCDF  = 0.79' in 'mbcsToSbcs': dot
## substituted for <81>

## Warning in text.default(x, y, ...): conversion failure on 'real CCDF  = 0.7'
## in 'mbcsToSbcs': dot substituted for <cf>

## Warning in text.default(x, y, ...): conversion failure on 'real CCDF  = 0.7'
## in 'mbcsToSbcs': dot substituted for <81>

## Warning in text.default(x, y, ...): conversion failure on 'Envelope CCDF  =
## 0.79' in 'mbcsToSbcs': dot substituted for <cf>

## Warning in text.default(x, y, ...): conversion failure on 'Envelope CCDF  =
## 0.79' in 'mbcsToSbcs': dot substituted for <81>
```

```
grid()
```