Hypoexponential_distribution_3links_test

2024-01-24

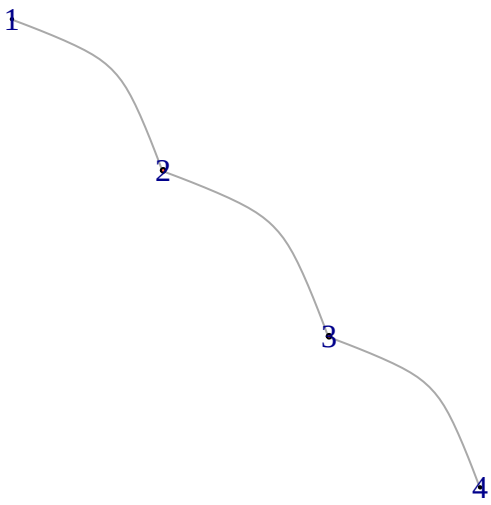
Input data

```
set.seed(123)
Capacity_Gbps = 10
Load = c(0.1, 0.4, 0.7)
N = 1250
```

Igraph theoretical calculations

```
## Warning: `get.edgelist()` was deprecated in igraph 2.0.0.
## i Please use `as_edgelist()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: `layout.auto()` was deprecated in igraph 2.0.0.
## i Please use `layout_nicely()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
E_x <- 8*N/(Capacity_Gbps*1e9)
theor_delay = (E_x*1/(1-Load))
print(paste("Link1 Kingman theor E(T) = ",  theor_delay[1], "s"))
```

```
## [1] "Link1 Kingman theor E(T) =  1.11111111111111e-06 s"
```

```
print(paste("Link2 Kingman theor E(T) = ",  theor_delay[2], "s"))
```

```
## [1] "Link2 Kingman theor E(T) =  1.66666666666667e-06 s"
```

```
print(paste("Link2 Kingman theor E(T) = ",  theor_delay[3], "s"))
```

```
## [1] "Link2 Kingman theor E(T) =  3.33333333333333e-06 s"
```

```
print(paste("Total Kingman theor E(T) = ",  sum(theor_delay), "s"))
```

```
## [1] "Total Kingman theor E(T) =  6.11111111111111e-06 s"
```

Simmer simulation

Delay results processing for each node

```
all_arrivals_res <- data.frame(env %>%
                                 get_mon_arrivals(per_resource = TRUE) %>%
                                 transform(waiting_time_in_queue = round(end_time - start_time - activi
                                 transform(spending_time = end_time - start_time))

for (node in 1:3){
  traffic <- dplyr::filter(all_arrivals_res, paste0("node_", node) == all_arrivals_res$resource)
  print(paste("Node - ", node, ":"))
  print(paste("simmer E(T) =", mean(traffic$spending_time), "s"))

}
```

```
## [1] "Node -  1 :"
## [1] "simmer E(T) = 1.11076358469178e-06 s"
## [1] "Node -  2 :"
## [1] "simmer E(T) = 1.71353997648868e-06 s"
## [1] "Node -  3 :"
## [1] "simmer E(T) = 3.3540214987292e-06 s"
```

Results of simulation e2e delay

```
all_arrivals_res <- data.frame(env %>%
                                 get_mon_arrivals(per_resource = FALSE) %>%
                                 transform(waiting_time_in_queue = round(end_time - start_time - activi
                                 transform(spending_time = end_time - start_time))

traffic <- dplyr::filter(all_arrivals_res, grepl("traffic_1_4_", all_arrivals_res$name))
print(mean(traffic$spending_time))
```

```
## [1] 6.175138e-06
```

```
delay_sim_mm1 <- traffic$spending_time
```

Calculations usind Hypoexponential formula

Calculate rates

```
E(g)$rate = (1 - E(g)$Load)/E_x
```

Calculate C1_3, C2_3, and C3_3

```
t = seq(0, 5e-5, 1e-8)

C1_3 = (E(g)$rate[2]/(E(g)$rate[2] - E(g)$rate[1])) * (E(g)$rate[3]/(E(g)$rate[3] - E(g)$rate[1]))
C2_3 = (E(g)$rate[1]/(E(g)$rate[1] - E(g)$rate[2])) * (E(g)$rate[3]/(E(g)$rate[3] - E(g)$rate[2]))
C3_3 = (E(g)$rate[1]/(E(g)$rate[1] - E(g)$rate[3])) * (E(g)$rate[2]/(E(g)$rate[2] - E(g)$rate[3]))
print(C1_3)
```

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

```
print(C2_3)
```

```
## [1] -3
```

```
print(C3_3)
```

```
## [1] 3
```

Calculate the individual PDFs for each link

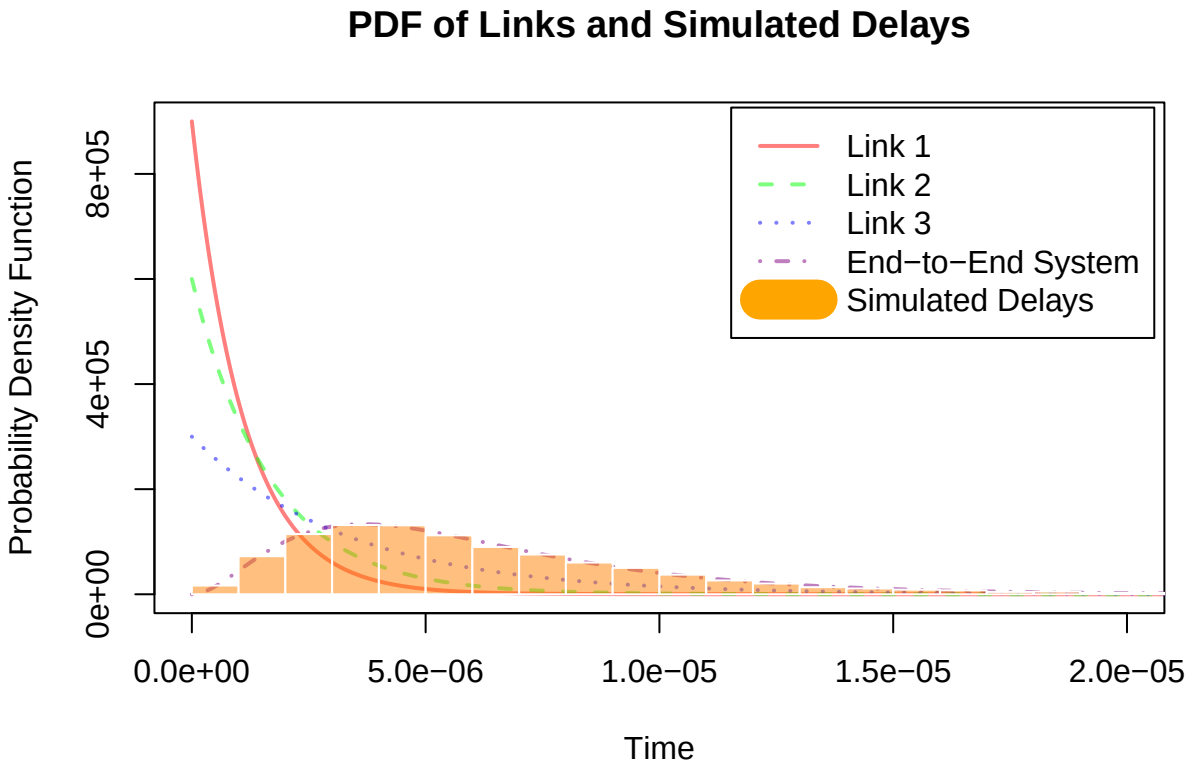
```
pdf_link1 = E(g)$rate[1] * exp(-E(g)$rate[1] * t)
pdf_link2 = E(g)$rate[2] * exp(-E(g)$rate[2] * t)
pdf_link3 = E(g)$rate[3] * exp(-E(g)$rate[3] * t)
```

Calculate the e2e PDF using the corrected rates

```
pdf_result = C1_3 * pdf_link1 + C2_3 * pdf_link2 + C3_3 * pdf_link3
```

Plot PDFs

```
plot(t, pdf_link1, type = "l", col = rgb(1, 0, 0, 0.5), lty = 1, lwd = 2,
     xlab = "Time", ylab = "Probability Density Function",
     main = "PDF of Links and Simulated Delays", xlim = c(0, 2e-5))

lines(t, pdf_link2, col = rgb(0, 1, 0, 0.5), lty = 2, lwd = 2)
lines(t, pdf_link3, col = rgb(0, 0, 1, 0.5), lty = 3, lwd = 2)

lines(t, pdf_result, col = rgb(0.5, 0, 0.5, 0.5), lty = 4, lwd = 2)

# Plot histogram of simulated delays with bandwidth and transparency
hist(delay_sim_mm1, prob = TRUE, col = rgb(1, 0.5, 0, 0.5), add = TRUE, breaks = 30, border = "white")

# Add legends for each component
legend("topright", legend=c("Link 1", "Link 2", "Link 3", "End-to-End System", "Simulated Delays"),
       col=c(rgb(1, 0, 0, 0.5), rgb(0, 1, 0, 0.5), rgb(0, 0, 1, 0.5), rgb(0.5, 0, 0.5, 0.5), "orange"),
       lty=c(1, 2, 3, 4, 1), lwd=c(2, 2, 2, 2, 20), inset = c(0.01, 0.01), xpd = TRUE)
```




```
# Reset to a single plot
par(mfrow=c(1, 1))
```

Calculation for n - number of links

```
calculate_C_i_n <- function(rate, i, n) {
  C_i_n <- 1
  for (j in 1:n) {
    if (j != i) {
      C_i_n <- C_i_n * (rate[j] / (rate[j] - rate[i]))
    }
  }
  return(C_i_n)
}

calculate_pdf_link <- function(rate, t) {
  return(rate * exp(-rate * t))
}

calculate_pdf_result <- function(rate, t, n) {
  pdf_result <- 0
  for (i in 1:n) {
    C_i_n <- calculate_C_i_n(rate, i, n)
    pdf_result <- pdf_result + C_i_n * calculate_pdf_link(rate[i], t)
  }
  return(pdf_result)
}
```

4 links

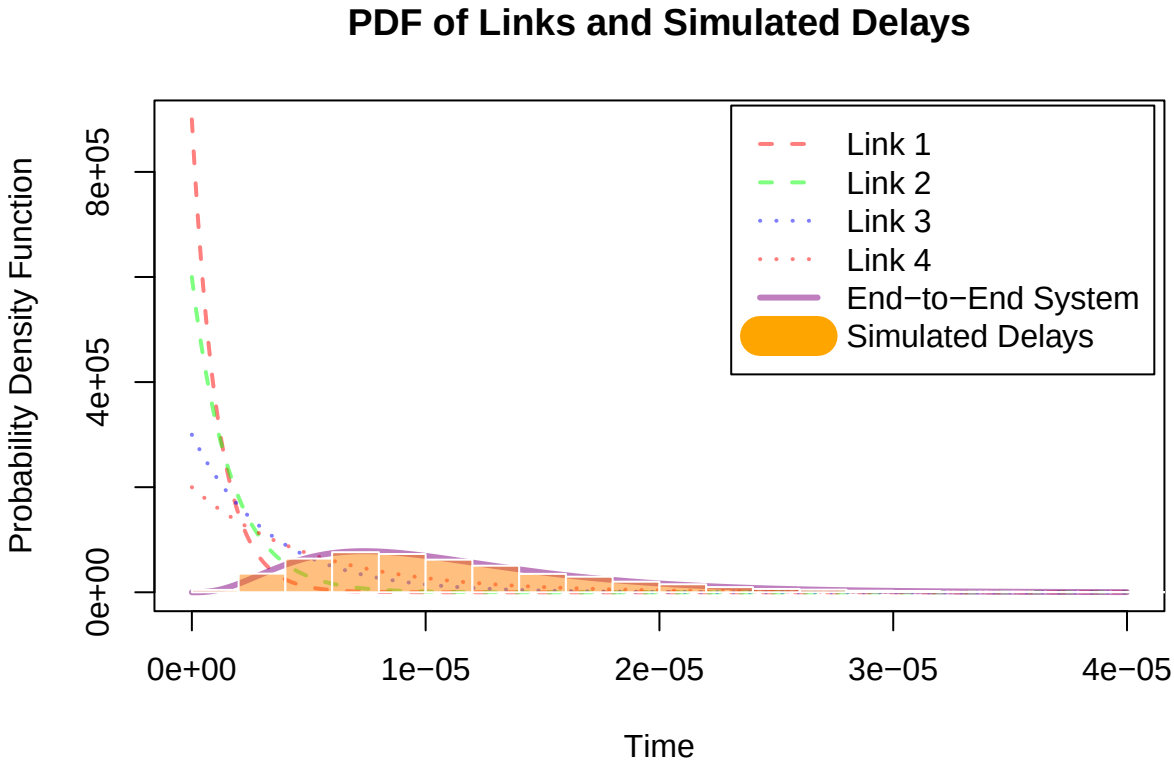
```
Load = c(0.1, 0.4, 0.7, 0.8)
```

```
print(mean(traffic$spending_time))
```

```
## [1] 1.137294e-05
```

```
delay_sim_mm1 <- traffic$spending_time
```

```
rate = (1 - Load)/E_x
t = seq(0, 4e-5, 1e-8)

pdf_result_4_links <- calculate_pdf_result(rate, t, 4)
pdf_link1 = rate[1] * exp(-rate[1] * t)
pdf_link2 = rate[2] * exp(-rate[2] * t)
pdf_link3 = rate[3] * exp(-rate[3] * t)
pdf_link4 = rate[4] * exp(-rate[4] * t)

# Set up a single plot with x-axis limits
plot(t, pdf_link1, type = "l", col = rgb(1, 0, 0, 0.5), lty = 2, lwd = 2,
     xlab = "Time", ylab = "Probability Density Function",
     main = "PDF of Links and Simulated Delays")

lines(t, pdf_link2, col = rgb(0, 1, 0, 0.5), lty = 2, lwd = 2)
lines(t, pdf_link3, col = rgb(0, 0, 1, 0.5), lty = 3, lwd = 2)
lines(t, pdf_link4, col = rgb(1, 0, 0, 0.5), lty = 3, lwd = 2)

lines(t, pdf_result_4_links, col = rgb(0.5, 0, 0.5, 0.5), lty = 1, lwd = 3)

# Plot histogram of simulated delays with less bandwidth and transparency
hist(delay_sim_mm1, prob = TRUE, col = rgb(1, 0.5, 0, 0.5), add = TRUE, breaks = 40, border = "white")

# Add legends for each component
legend("topright", legend=c("Link 1", "Link 2", "Link 3","Link 4", "End-to-End System", "Simulated Dela
       col=c(rgb(1, 0, 0, 0.5), rgb(0, 1, 0, 0.5), rgb(0, 0, 1, 0.5),rgb(1, 0, 0, 0.5), rgb(0.5, 0, 0.5
       lty=c(2, 2, 3, 3, 1,  1), lwd=c(2, 2, 2, 2, 3, 20), inset = c(0.01, 0.01), xpd = TRUE)
```




```
# Reset to a single plot
par(mfrow=c(1, 1))
```