213 Mid Two Tellers

Zixi Zhang

2022-11-09

1) Conduct a simulation on your Urgent Care Center business.

Here are some simplifying assumptions:

Assume that your business opens at 8am and closes at 5pm. Customers arrive at a rate of 4/hr. Between the hours of noon and 1 that rate increases to 6/hr. Suppose you have 2 tellers. There are, in general, 3 types of queuing systems 1) 1 line with parallel servers. 2) tandem service (In-N-out example) and 3) where each teller has their own line (grocery store).

What is the expected waiting time for each type of system. Simulate 1 week (do 7 times and take the average). Which system do you think is the best? Does it matter where you place the "slow" teller? Does you "optimal" depend on your choices?

Setting up Function

```
lambda.t <- function(time) {
    # rate increases to 6/hr between the hours of noon and 1
    if (time >= 4 & time <= 5) {
      lambda <- 6
    # Customers arrive at a rate of 4/hr  in less busy time
    } else {
      lambda <- 4
    }
    list(lamt = lambda)
}

inter_arrival_time <- function(n = 1,t =0){
  u <- runif(n)
  x <- -log(u) / lambda.t(t)$lamt
  x
}
```

Simulating an exponential random variable with parameter lambda to generate service completion time:

```
expsim <- function(n = 1 , lambda = 1){
  u <- runif(n)
  x <- -log(u) / lambda
  x
}
```

medium: all servers have different rates. Note that tandem should be different than the other 2 (think about why in-n-out has tandem service)

One line two parallel servers

Server 1 ~ serves 5 customers per hour (12 minutes per customer)

Server 2 ~ serves 3 customers per hour (20 minutes per customer)

```
#reset Time past T and waiting time
Tp <- -1
meanminwait <- -1

set.seed(159)

for (i in 1:7) {
  t <- 0 # time
  Na <- 0 # total number of arrivals at time t
  Nd <- 0 # total number of departures at time t
  i1 <- 0 # customer being served by server 1
  i2 <- 0 # customer being served by server 2
  c1 <- 0 # total number of customers that are served by server 1
  c2 <- 0 # total number of customers that are served by server 2
  n <- 0 # number of customers in store at time t
  A <- 0 # vector of arrival times of each customer
  D <- 0 # vector of departure times of each customer
  Tp[i] <- -1 # time past closing that the last customer departs
  ta <- inter_arrival_time(t = 0)# time of first arrival
  t1 <- 99999 # time of next departure from server 1
  t2 <- 99999 # time of next departure from server 2

  while (Tp[i] == -1) {
    # arrival occurs before a departure occurs
    if (ta == min(ta, t1, t2) & (ta < 9) ) {
      t <- ta
      Na <- Na + 1
      X <- inter_arrival_time(t = ta) # generating next inter-arrival time
      ta <- X + t # setting next arrival time
      A[Na] <- t # recording the time of the current arrival
      if (n == 0 | (n == 1 & i1 == 0)) {
        n <- n + 1
        i1 <- Na
        Y1 <- expsim(lambda = 5) # generating service time
        t1  <- t + Y1 # generating next departure time
      } else if (n == 1 & i2 == 0) {
        n <- n + 1
        i2 <- Na
        Y2 <- expsim(lambda = 3)
        t2 <- t + Y2
      } else if (n >= 2) {
        n <- n + 1
      }

    # someone departs from server 1
    } else if ((t1 < ta & t1 <= t2) | (ta > 9 & t1 <= t2 & n>0)) {
      t <- t1
      c1 <- c1 + 1
      Nd <- Nd + 1
```

```
      D[i1] <- t
      if (n == 1){
        n <- 0
        i1 <- 0
        t1 <- 99999
      } else if (n==2) {
        n <- n - 1
        i1 <- 0
        t1 <- 99999
      } else if (n > 2) {
        n <- n - 1
        i1 <- max(i1, i2) + 1
        Y1 <- expsim(lambda = 5)
        t1 <- t + Y1
      }
    # someone departs from server 2
    } else if ((t2 < ta & t2 < t1) | (ta > 9 & t1 > t2 & n > 0)) {
      t <- t2
      c2 <- c2 + 1
      Nd <- Nd + 1
      D[i2] <- t
      if (n == 1 | n == 2){
        n <- n-1
        i2 <- 0
        t2 <- 99999
      } else if (n > 2) {
        n <- n - 1
        i2 <- max(i1, i2) + 1
        Y2 <- expsim(lambda = 3)
        t2 <- t + Y2
      }
    } else if (n == 0 & ta > 9){
      Tp[i] <- (max(D) - 9)*60
    }
    }
  wait <- D - A
  meanminwait[i] <- mean(wait)*60
}

Tp[Tp<0] <- 0
```

```
result1 <- list(mean(meanminwait),mean(Tp))
names(result1) <- c("Parallel: Average waiting time of a customer",
"Parallel: The average time past T last customer leaves TP")
print(result1)
```

```
## $`Parallel: Average waiting time of a customer`
## [1] 17.21876
##
## $`Parallel: The average time past T last customer leaves TP`
## [1] 16.90438
```

The average waiting time of a customer is 17.22 minutes, and the average time of last customer leaving the

store is 16.90 minutes after closing.

Two servers in tandem

In the one line two parallel servers system, there was one server that took about 12 minutes per customer, and another server that took about 20 minutes per customer. So on average, the service time took about 16 minutes per customer.

Here, set the first server's service time to be about 6 minutes per customer (can serve 10 customers per hour). Also set the second server's service time to be about 10 minutes per customer (can serve 6 customers per hour). This means that on average the total service time will be 16 minutes which is same to what it took in the previous two server case.

Server 1 ~ serves 10 customers per hour (6 minutes per customer)

Server 2 ~ serves 6 customers per hour (10 minutes per customer)

```
Tp <- -1
meanminwait <- -1
set.seed(357)
# run simulation for a week
for (i in 1:7) {
  # setting initial values
  A1 <- 0 # vector of arrival times to server 1
  A2 <- 0 # vector of arrival times to server 2
  t <- 0 # current time
  t1 <- 99999 # departure time from server 1
  t2 <- 99999 # departure time from server 2
  n1 <- 0 # number of people in line at server 1
  n2 <- 0 # number of people in line at server 2
  D <- 0 # departure time vector
  Na <- 0 # total number of arrivals
  Nd <- 0 # total number of departures
  ta <- inter_arrival_time(t = 0) # first arrival time
  Tp[i] <- -1 # will be the time after closing the last customer leaves

# condition while loop to run until last person leaves
  while (Tp[i] == -1) {
    # someone arrives before closing
    if (ta == min(ta, t1, t2) & ( ta<9)) {
      t <- ta # update time to current arrival time
      n1 <- n1 + 1 # update number of people in line for server 1
      Na <- Na + 1 # update total number of arrivals
      X <- inter_arrival_time(t = ta)
      ta <- t + X # next arrival time
      A1[Na] <- t # record arrival time
      if (n1 == 1) {
        Y1 <- expsim(lambda = 10)
        t1 <- t + Y1 # next departure time from server 1
      }
    # someone departs from server 1 and goes to server 2 line
    } else if((t1 < ta & t1 <= t2) | (ta > 9 & n1 > 0)) {
      t <- t1
      n1 <- n1 - 1
      n2 <- n2 + 1
```

```
      A2[Na - n1] <- t
      if (n1 == 0) {
        t1 <- 99999
      } else {
        Y1 <- expsim(lambda = 10)
        t1 <- t + Y1
      }
      if (n2 == 1) {
        Y2 <- expsim(lambda = 6)
        t2 <- t + Y2
      }
    # someone leaves from server 2 (leaves store)
    } else if (t2 < ta & t2 < t1 | (ta > 9 & t2 < t1)) {
      t <- t2
      Nd <- Nd + 1
      n2 <- n2 - 1
      D[Nd] <- t
      if (n2 == 0) {
        t2 <- 99999
      } else if (n2 > 0) {
        Y2 <- expsim(lambda = 6)
        t2 <- t + Y2
      }
    # if no one is in the store and it's after closing, record time of last departure
    } else if (ta > 9 & (n1+n2) == 0) {
      Tp[i] <- (max(D) - 9)*60
    }
  }
  # wait times for each customer (in hours):
  wait <- D - A1
  # mean wait times in minutes for each customer for day i
  meanminwait[i] <- mean(wait)*60
}

Tp[Tp<0] <- 0
```

```
result2 <- list(mean(meanminwait),mean(Tp))
names(result2) <- c("Tandem: Average waiting time of a customer",
                    "Tandem: The average time past T last customer leaves TP")
print(result2)
```

```
## $`Tandem: Average waiting time of a customer`
## [1] 35.30783
##
## $`Tandem: The average time past T last customer leaves TP`
## [1] 29.682
```

The average waiting time of a customer is 35.31 minutes, and the average time of last customer leaving the store is 29.68 minutes after closing.

Each server has their own line

Assume that the person comes into the store will choose that line that is shorter, and if the lines are the same length will default to choosing line 1.

Server 1 ~ serves 5 customers per hour (12 minutes per customer)

Server 2 ~ serves 3 customers per hour (20 minutes per customer)

Same as one line two parallel servers system

```
Tp <- -1
meanminwait <- -1
set.seed(304)
# run simulation for a week
for (i in 1:7) {
  # initial values
  n1 <- 0 # number of people in line 1
  n2 <- 0 # number of people in line 2
  A1 <- 0 # vector of times of arrivals to line 1
  A2 <- 0 # vector of times of arrivals to line 2
  D1 <- 0 # vector of times of departures from line 1
  D2 <- 0 # vector of times of departures from line 2
  Na1 <- 0 # total number of arrivals to line 1
  Na2 <- 0 # total number of arrivals to line 2
  Nd1 <- 0 # total number of departures from line 1
  Nd2 <- 0 # total number of departures from line 2
  t <- 0 # current time
  t1 <- 99999 # time of next departure from line 1
  t2 <- 99999 # time of next departure from line 2
  ta <- inter_arrival_time(t = 0) # time of first arrival
  Tp[i] <- -1

  while (Tp[i] == -1) {
    # arrival while line 1 is shorter
    if (n1 <= n2 & ta < min(t1, t2) & (ta < 9)) {
      t <- ta
      n1 <- n1 + 1
      X <- inter_arrival_time(t = ta)
      ta <- t + X
      Na1 <- Na1 + 1
      A1[Na1] <- t
      if (n1 == 1) {
        Y1 <- expsim(lambda = 5)
        t1 <- t + Y1
      }
    # arrival while line 2 is shorter
    } else if (n2 < n1 & ta < min(t1, t2) & (ta < 9)) {
      t <- ta
      n2 <- n2 + 1
      X <- inter_arrival_time(t = ta)
      ta <- t + X
      Na2 <- Na2 + 1
      A2[Na2] <- t
      if (n2 == 1){
        Y2 <- expsim(lambda = 3)
```

```
        t2 <- t + Y2
      }
    # departure from line 1
    } else if ((t1 < ta & t1 <= t2) | (ta > 9 & t1 <= t2 & (n1 + n2) > 0)) {
      t <- t1
      n1 <- n1 - 1
      Nd1 <- Nd1 + 1
      D1[Nd1] <- t
      if (n1 > 0) {
        Y1 <- expsim(lambda = 5)
        t1 <- t + Y1
      } else if (n1 == 0) {
        t1 <- 99999
      }
    # departure from line 2
    } else if ((t2 < ta & t2 < t1) | (ta > 9 & t2 < t1 & (n1 + n2) > 0))  {
      t <- t2
      n2 <- n2 - 1
      Nd2 <- Nd2 + 1
      D2[Nd2] <- t
      if (n2 > 0){
        Y2 <- expsim(lambda = 3)
        t2 <- t + Y2
      } else if (n2 == 0) {
        t2 = 99999
      }
    } else if (ta > 9 & (n1 + n2) == 0) {
      Tp[i] <- (max(D1, D2) - 9)*60
      wait <- c(D1 - A1, D2 - A2)
      meanminwait[i] <- mean(wait)*60
    }
  }
}

Tp[Tp<0] <- 0
```

```
result3 <- list(mean(meanminwait),mean(Tp))
names(result3) <- c("Grocery store: Average waiting time of a customer",
                    "Grocery store: The average time past T last customer leaves TP")
print(result3)
```

```
## $`Grocery store: Average waiting time of a customer`
## [1] 20.85796
##
## $`Grocery store: The average time past T last customer leaves TP`
## [1] 17.4259
```

The average waiting time of a customer is 20.86 minutes, and the average time of last customer leaving the store is 17.43 minutes after closing.

Conclusion

```
do.call(c, list(result1,result2,result3))
```

```
## $`Parallel: Average waiting time of a customer`
## [1] 17.21876
##
## $`Parallel: The average time past T last customer leaves TP`
## [1] 16.90438
##
## $`Tandem: Average waiting time of a customer`
## [1] 35.30783
##
## $`Tandem: The average time past T last customer leaves TP`
## [1] 29.682
##
## $`Grocery store: Average waiting time of a customer`
## [1] 20.85796
##
## $`Grocery store: The average time past T last customer leaves TP`
## [1] 17.4259
```

We can see the one line two parallel servers system has the shortest average waiting time of 17.22 minutes and the last customer leaves at an average time of 16.90 minutes after closing(which is also the shortest among the three systems).

Try place the "slow" teller as server1

One line two parallel servers

Server 1 ~ serves 3 customers per hour (20 minutes per customer)

Server 2 ~ serves 5 customers per hour (12 minutes per customer)

```
#reset Time past T and waiting time
Tp <- -1
meanminwait <- -1

set.seed(159)

for (i in 1:7) {
  t <- 0 # time
  Na <- 0 # total number of arrivals at time t
  Nd <- 0 # total number of departures at time t
  i1 <- 0 # customer being served by server 1
  i2 <- 0 # customer being served by server 2
  c1 <- 0 # total number of customers that are served by server 1
  c2 <- 0 # total number of customers that are served by server 2
  n <- 0 # number of customers in store at time t
  A <- 0 # vector of arrival times of each customer
  D <- 0 # vector of departure times of each customer
  Tp[i] <- -1 # time past closing that the last customer departs
```

```
ta <- inter_arrival_time(t = 0)# time of first arrival
t1 <- 99999 # time of next departure from server 1
t2 <- 99999 # time of next departure from server 2

while (Tp[i] == -1) {
  # arrival occurs before a departure occurs
  if (ta == min(ta, t1, t2) & (ta < 9) ) {
    t <- ta
    Na <- Na + 1
    X <- inter_arrival_time(t = ta) # generating next inter-arrival time
    ta <- X + t # setting next arrival time
    A[Na] <- t # recording the time of the current arrival
    if (n == 0 | (n == 1 & i1 == 0)) {
      n <- n + 1
      i1 <- Na
      Y1 <- expsim(lambda = 3) # generating service time
      t1  <- t + Y1 # generating next departure time
    } else if (n == 1 & i2 == 0) {
      n <- n + 1
      i2 <- Na
      Y2 <- expsim(lambda = 5)
      t2 <- t + Y2
    } else if (n >= 2) {
      n <- n + 1
    }

  # someone departs from server 1
  } else if ((t1 < ta & t1 <= t2) | (ta > 9 & t1 <= t2 & n>0)) {
    t <- t1
    c1 <- c1 + 1
    Nd <- Nd + 1
    D[i1] <- t
    if (n == 1){
      n <- 0
      i1 <- 0
      t1 <- 99999
    } else if (n==2) {
      n <- n - 1
      i1 <- 0
      t1 <- 99999
    } else if (n > 2) {
      n <- n - 1
      i1 <- max(i1, i2) + 1
      Y1 <- expsim(lambda = 3)
      t1 <- t + Y1
    }
  # someone departs from server 2
  } else if ((t2 < ta & t2 < t1) | (ta > 9 & t1 > t2 & n > 0)) {
    t <- t2
    c2 <- c2 + 1
    Nd <- Nd + 1
    D[i2] <- t
    if (n == 1 | n == 2){
```

```
        n <- n-1
        i2 <- 0
        t2 <- 99999
      } else if (n > 2) {
        n <- n - 1
        i2 <- max(i1, i2) + 1
        Y2 <- expsim(lambda = 5)
        t2 <- t + Y2
      }
    } else if (n == 0 & ta > 9){
      Tp[i] <- (max(D) - 9)*60
    }
    }
  wait <- D - A
  meanminwait[i] <- mean(wait)*60
}

Tp[Tp<0] <- 0

result4 <- list(mean(meanminwait),mean(Tp))
names(result4) <- c("Parallel reversed: Average waiting time of a customer",
"Parallel reversed: The average time past T last customer leaves TP")
do.call(c, list(result1,result4))
```

```
## $`Parallel: Average waiting time of a customer`
## [1] 17.21876
##
## $`Parallel: The average time past T last customer leaves TP`
## [1] 16.90438
##
## $`Parallel reversed: Average waiting time of a customer`
## [1] 20.164
##
## $`Parallel reversed: The average time past T last customer leaves TP`
## [1] 31.26474
```

For parallel system, placing the slower teller as server1 increases the average waiting time and average time that last customer leaves after closing.

Two servers in tandem

Server 1 ~ serves 6 customers per hour (10 minutes per customer)

Server 2 ~ serves 10 customers per hour (6 minutes per customer)

```
Tp <- -1
meanminwait <- -1
set.seed(357)
# run simulation for a week
for (i in 1:7) {
  # setting initial values
  A1 <- 0 # vector of arrival times to server 1
  A2 <- 0 # vector of arrival times to server 2
```

```
  t <- 0 # current time
  t1 <- 99999 # departure time from server 1
  t2 <- 99999 # departure time from server 2
  n1 <- 0 # number of people in line at server 1
  n2 <- 0 # number of people in line at server 2
  D <- 0 # departure time vector
  Na <- 0 # total number of arrivals
  Nd <- 0 # total number of departures
  ta <- inter_arrival_time(t = 0) # first arrival time
  Tp[i] <- -1 # will be the time after closing the last customer leaves

# condition while loop to run until last person leaves
  while (Tp[i] == -1) {
    # someone arrives before closing
    if (ta == min(ta, t1, t2) & ( ta<9)) {
      t <- ta # update time to current arrival time
      n1 <- n1 + 1 # update number of people in line for server 1
      Na <- Na + 1 # update total number of arrivals
      X <- inter_arrival_time(t = ta)
      ta <- t + X # next arrival time
      A1[Na] <- t # record arrival time
      if (n1 == 1) {
        Y1 <- expsim(lambda = 6)
        t1 <- t + Y1 # next departure time from server 1
      }
    # someone departs from server 1 and goes to server 2 line
    } else if((t1 < ta & t1 <= t2) | (ta > 9 & n1 > 0)) {
      t <- t1
      n1 <- n1 - 1
      n2 <- n2 + 1
      A2[Na - n1] <- t
      if (n1 == 0) {
        t1 <- 99999
      } else {
        Y1 <- expsim(lambda = 6)
        t1 <- t + Y1
      }
      if (n2 == 1) {
        Y2 <- expsim(lambda = 10)
        t2 <- t + Y2
      }
    # someone leaves from server 2 (leaves store)
    } else if (t2 < ta & t2 < t1 | (ta > 9 & t2 < t1)) {
      t <- t2
      Nd <- Nd + 1
      n2 <- n2 - 1
      D[Nd] <- t
      if (n2 == 0) {
        t2 <- 99999
      } else if (n2 > 0) {
        Y2 <- expsim(lambda = 10)
        t2 <- t + Y2
      }
```

```
    # if no one is in the store and it's after closing, record time of last departure
    } else if (ta > 9 & (n1+n2) == 0) {
      Tp[i] <- (max(D) - 9)*60
    }
  }
  # wait times for each customer (in hours):
  wait <- D - A1
  # mean wait times in minutes for each customer for day i
  meanminwait[i] <- mean(wait)*60
}

Tp[Tp<0] <- 0

result5 <- list(mean(meanminwait),mean(Tp))
names(result5) <- c("Tandem reversed: Average waiting time of a customer",
                    "Tandem reversed: The average time past T last customer leaves TP")
do.call(c, list(result2,result5))
```

```
## $`Tandem: Average waiting time of a customer`
## [1] 35.30783
##
## $`Tandem: The average time past T last customer leaves TP`
## [1] 29.682
##
## $`Tandem reversed: Average waiting time of a customer`
## [1] 32.93784
##
## $`Tandem reversed: The average time past T last customer leaves TP`
## [1] 14.5195
```

For parallel system, placing the slower teller as server1 reduces both the average waiting time and average time that last customer leaves after closing. The reduction of average time that the last customer leaves after closing is significant.

Each server has their own line

Server 1 ~ serves 3 customers per hour (20 minutes per customer)

Server 2 ~ serves 5 customers per hour (12 minutes per customer)

```
Tp <- -1
meanminwait <- -1
set.seed(304)
# run simulation for a week
for (i in 1:7) {
  # initial values
  n1 <- 0 # number of people in line 1
  n2 <- 0 # number of people in line 2
  A1 <- 0 # vector of times of arrivals to line 1
  A2 <- 0 # vector of times of arrivals to line 2
  D1 <- 0 # vector of times of departures from line 1
  D2 <- 0 # vector of times of departures from line 2
  Na1 <- 0 # total number of arrivals to line 1
```

```
Na2 <- 0 # total number of arrivals to line 2
Nd1 <- 0 # total number of departures from line 1
Nd2 <- 0 # total number of departures from line 2
t <- 0 # current time
t1 <- 99999 # time of next departure from line 1
t2 <- 99999 # time of next departure from line 2
ta <- inter_arrival_time(t = 0) # time of first arrival
Tp[i] <- -1

while (Tp[i] == -1) {
  # arrival while line 1 is shorter
  if (n1 <= n2 & ta < min(t1, t2) & (ta < 9)) {
    t <- ta
    n1 <- n1 + 1
    X <- inter_arrival_time(t = ta)
    ta <- t + X
    Na1 <- Na1 + 1
    A1[Na1] <- t
    if (n1 == 1) {
      Y1 <- expsim(lambda = 3)
      t1 <- t + Y1
    }
  # arrival while line 2 is shorter
  } else if (n2 < n1 & ta < min(t1, t2) & (ta < 9)) {
    t <- ta
    n2 <- n2 + 1
    X <- inter_arrival_time(t = ta)
    ta <- t + X
    Na2 <- Na2 + 1
    A2[Na2] <- t
    if (n2 == 1){
      Y2 <- expsim(lambda = 5)
      t2 <- t + Y2
    }
  # departure from line 1
  } else if ((t1 < ta & t1 <= t2) | (ta > 9 & t1 <= t2 & (n1 + n2) > 0)) {
    t <- t1
    n1 <- n1 - 1
    Nd1 <- Nd1 + 1
    D1[Nd1] <- t
    if (n1 > 0) {
      Y1 <- expsim(lambda = 3)
      t1 <- t + Y1
    } else if (n1 == 0) {
      t1 <- 99999
    }
  # departure from line 2
  } else if ((t2 < ta & t2 < t1) | (ta > 9 & t2 < t1 & (n1 + n2) > 0))  {
    t <- t2
    n2 <- n2 - 1
    Nd2 <- Nd2 + 1
    D2[Nd2] <- t
    if (n2 > 0){
```

```
        Y2 <- expsim(lambda = 5)
        t2 <- t + Y2
      } else if (n2 == 0) {
        t2 = 99999
      }
    } else if (ta > 9 & (n1 + n2) == 0) {
      Tp[i] <- (max(D1, D2) - 9)*60
      wait <- c(D1 - A1, D2 - A2)
      meanminwait[i] <- mean(wait)*60
    }
  }
}

Tp[Tp<0] <- 0

result6 <- list(mean(meanminwait),mean(Tp))
names(result6) <- c("Grocery store reversed: Average waiting time of a customer",
                    "Grocery store reversed: The average time past T last customer leaves TP")
do.call(c, list(result3,result6))
```

```
## $`Grocery store: Average waiting time of a customer`
## [1] 20.85796
##
## $`Grocery store: The average time past T last customer leaves TP`
## [1] 17.4259
##
## $`Grocery store reversed: Average waiting time of a customer`
## [1] 21.76622
##
## $`Grocery store reversed: The average time past T last customer leaves TP`
## [1] 22.34298
```

For grocery store system, placing the slower teller as server1 increases the average waiting time and average time that the last customer leaves after closing.

Conclusion for reversed systems

```
do.call(c, list(result4,result5,result6))
```

```
## $`Parallel reversed: Average waiting time of a customer`
## [1] 20.164
##
## $`Parallel reversed: The average time past T last customer leaves TP`
## [1] 31.26474
##
## $`Tandem reversed: Average waiting time of a customer`
## [1] 32.93784
##
## $`Tandem reversed: The average time past T last customer leaves TP`
## [1] 14.5195
##
```

```
## $`Grocery store reversed: Average waiting time of a customer`
## [1] 21.76622
##
## $`Grocery store reversed: The average time past T last customer leaves TP`
## [1] 22.34298
```

After placing the slower teller as server1, the parallel remains to be the best system.

Overall Conclusion

```
do.call(c, list(result1,result2,result3,result4,result5,result6))
```

```
## $`Parallel: Average waiting time of a customer`
## [1] 17.21876
##
## $`Parallel: The average time past T last customer leaves TP`
## [1] 16.90438
##
## $`Tandem: Average waiting time of a customer`
## [1] 35.30783
##
## $`Tandem: The average time past T last customer leaves TP`
## [1] 29.682
##
## $`Grocery store: Average waiting time of a customer`
## [1] 20.85796
##
## $`Grocery store: The average time past T last customer leaves TP`
## [1] 17.4259
##
## $`Parallel reversed: Average waiting time of a customer`
## [1] 20.164
##
## $`Parallel reversed: The average time past T last customer leaves TP`
## [1] 31.26474
##
## $`Tandem reversed: Average waiting time of a customer`
## [1] 32.93784
##
## $`Tandem reversed: The average time past T last customer leaves TP`
## [1] 14.5195
##
## $`Grocery store reversed: Average waiting time of a customer`
## [1] 21.76622
##
## $`Grocery store reversed: The average time past T last customer leaves TP`
## [1] 22.34298
```

Comparing all the possible systems, we can see parallel system that sets the faster teller as server1 has the shortest average waiting time, while tandem system that sets the slower teller as server1 has the shortest average time that the last customer leaves after closing. It seems that setting faster teller as server1 is better for both parallel and grocery store systems, while setting slower teller as server1 is better for tandem system.

Overall, parallel system that sets the faster teller as server1 is the best system.