Network Management Project

Jiannan Guo - ICT-Innovation
Niklas Semmler - ICT-Innovation

2. Dezember 2013

1. Task

1.1. Pseudocode

The following pseudo code (listing 1) is inspired by [1]. To the existing object A a new object Comm was introduced to handle all message exchange.

Listing 1: Pseudocode for $\widetilde{f}(t)$ with rate control

```
1  messages:
2  (ResponseTimeArriveMessage, responseTime) # similar to local
3  (UpdateVector, neighbor) # similar to new and update
4  (TimeOut, responseTime)
5
6  objects and help functions:
7  A       # storing values for and performing aggregation
8  Comm    # communication messages
9  A.aggregate_local       # aggregate local response times for local statistics
10 A.aggregate_sub_tree    # combine local values with children's for statistics
11                         #  of the whole subtree
12 A.value                 # values of statistics of subtree (including own)
13 Comm.send_to_self       # send a message to the own node with a delay
14 Comm.send_to_neighbors  # send to all specified neighbors
15 Comm.send_to_parent     # send to parent
16 find_parent             # among neighbors, choose first with minimum level as
17                         #  new parent
18 constants:
19 DELAY := # fixed time window
20 INF := # really big number
21
22 global variables:
23 msgBudget = 1
24
25 procedure GAP( )
26     Neighbors := empty;
27     ResponseTimes := empty;
28
29     if v = root then
30         level = 0
```

```
31          parent = 0
32      else
33          level = INF
34          parent = INF
35      end if
36
37      A.initiate()
38      while true do
39          read message;
40          switch (message)
41              case (ResponseTimeArriveMessage, responseTime)
42                  ResponseTimes.add(responseTime)
43                  # to keep only a set of responseTimes in cache
44                  Comm.send_to_self(DELAY, TimeOut(responseTime))
45                  A.aggregate_local(ResponseTimes)
46                  A.aggregate_sub_tree(Neighbors)
47              case (UpdateVector, neighbor):
48                  Neighbors.add(neighbor)
49                  (newParent, newLevel) = find_new_parent(Neighbors)
50                  A.aggregate_sub_tree(Neighbors)
51                  if (level != newLevel || parent != newParent) then
52                      level = newLevel
53                      parent = newParent
54                      Comm.send_to_neighbors(Neighbors, level, parent, A.value())
55                      continue
56                  end if
57              # to keep only a set of responseTimes in cache
58              case (TimeOut, responseTime):
59                  responses.remove(responseTime)
60                  A.aggregate_local_value(ResponseTimes)
61                  A.aggregate_sub_tree(Neighbors)
62          end switch
63          if (msgBudget > 0) then
64              Comm.send_to_parent(Neighbors, A.value)
65              msgBudget--
66          end if
67          if (msgBudgetReset) then # waiting for another program to change
68              msgBudget = MSG_BUDGET
69          endif
70      end while
71  end procedure
72  procedure ResetMsgBudget()
73      msgBudget = 1
74  end procedure
```

1.2. Implementation Details

A new base class was implemented which all of the tasks extend: `peersim.EP2300.vector.GAPNode`. In `peersim.EP2300.message` three new messages were introduced: `ResponseTimeArriveMessage`, `UpdateVector`, `TimeOut`. The first two roughly equal `LOCALVAR` and `UPDATE & NEW` from the original technical report (see [1]).

Each node stores all its information on it's neighbors as `NodeStateVector` in a SortMap named `neighborList`. Details on the nodes response times are stored in the ArrayList

`requestList`. To focus only on the response times in the current time window, every response time is assigned a time out.

For the implementation with rate control, every message send to the parent will trigger a decrease of the msg budget. As soon as the msg budget is lower or equal to 0, no more messages are sent out. The message budget is reset via a new control `peersim.EP2300.control.ResetMsgBudget`.

1.3. Results

(i) time series of f(t) and f(t) for r = {0.2,0.4,0.8} and {R1} from the first 5 minutes

(ii) time series of f(t) and f(t) for r = {0.2,0.4,0.8} and {R1} after the first 5 minutes

(iii) trade-off plots for both {R1,R2} (and all rate options?? 0.1,0.2,0.4,0.8,1.6)

(iv) density plots for r = {0.2, 0.4, 0.8} and {R1,R2}

1.4. plots

2. Task

2.1. Pseudocode

In the following the pseudo code for extension 1 is presented. Different to the first task we do not use a message budget (or message rate). Instead an error budget is used. This error budget is assigned to all nodes. Only when the difference of the aggregate of the subtree to the last communicated version of the same exceeds the error budget, a message is sent.

Listing 2: Pseudocode for $\widetilde{f}(t)$ with error budget in P_1

```
1  messages:
2  [see listing 1]
3
4  objects and help functions:
5  [see listing 1]
6  A.difference_of_subtree # difference of last communicated value versus current
7                          #  communicated value
8
9  constants:
10 DELAY := # fixed time window
11 INF := # really big number
12 ERROR_BUDGET = # fix number
```

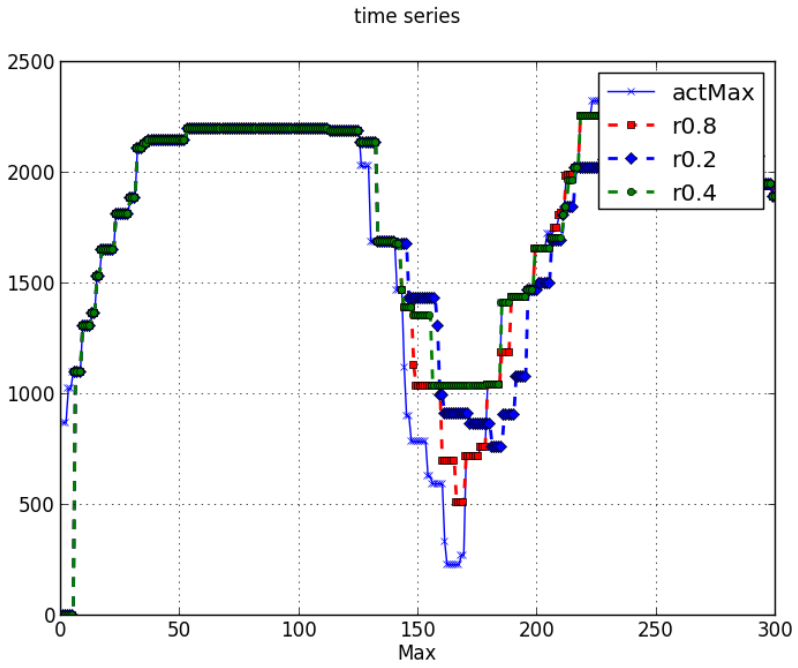

Abbildung 1: Time series for pattern R1

```
13
14  global variables:
15
16  procedure GAP( )
17      Neighbors := empty;
18      ResponseTimes := empty;
19
20      if v = root then
21          [see listing 1]
22      end if
23
24      A.initiate()
25      while true do
26          read message;
27          switch (message)
28              [see listing 1]
29          end switch
30          if (A.difference_of_subtree() > ERROR_BUDGET) then
31              Comm.send_to_parent(Neighbors, A.value)
32          end if
33      end while
34  end procedure
```

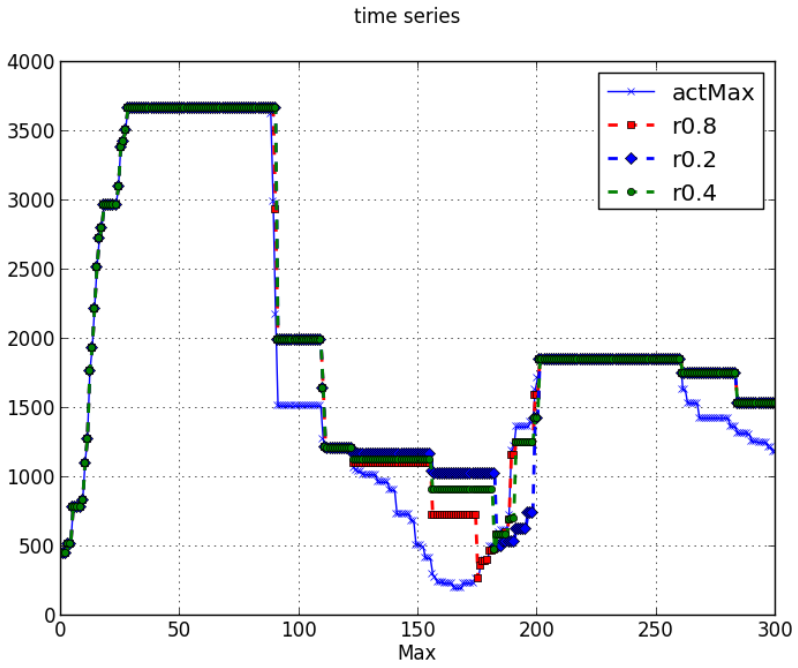

Abbildung 2: Time series for pattern R2

2.2. Implementation Details

Similar to the difference between the pseudocode of this and task 1 the difference in the code is just the replacement of a depletable message budget with a constant error budget.

2.3. plots

3. Task

3.1. Pseudocode

The pseudocode for providing $\widetilde{g}(t)$ under P_1 is equal to the one provided in section 3. The pseudocode for P_2 is attached below.

Listing 3: Pseudocode for $\widetilde{f}(t)$ with error budget in P_1

```
1 messages:
2 [see listing 1]
3 (ErrorBudget, errorBudget)
```

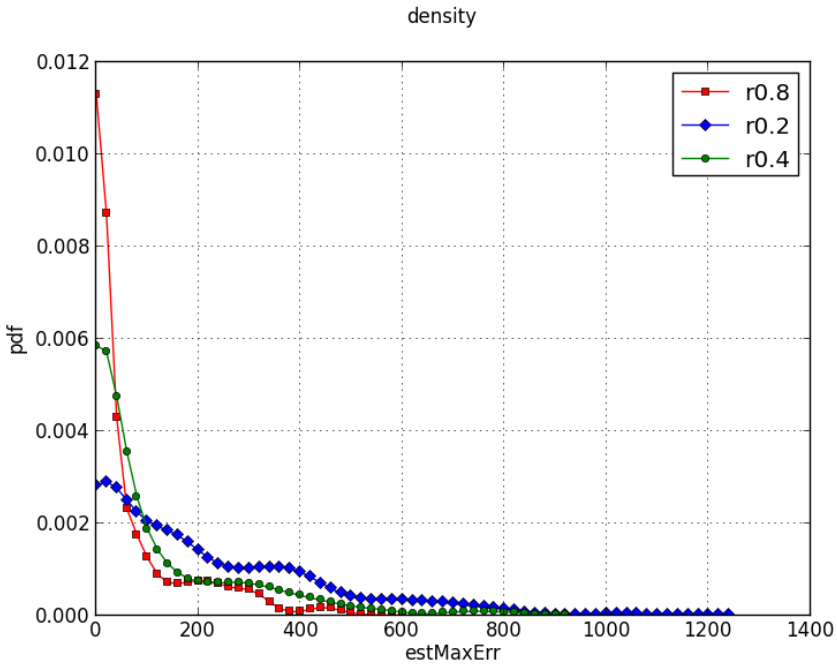

Abbildung 3: Density plot for pattern R1

```
4
5   objects and help functions:
6   [see listing 1]
7   A.difference_of_subtree # difference of last communicated value versus current
8                           #  communicated value
9
10  constants:
11  DELAY := # fixed time window
12  INF := # really big number
13  ERROR_BUDGET = # fix number
14
15  global variables:
16
17  procedure GAP( )
18      Neighbors := empty;
19      ResponseTimes := empty;
20
21      if v = root then
22          [see listing 1]
23      end if
24
25      A.initiate()
26      while true do
27          read message;
28          switch (message)
29              [see listing 1]
30          end switch
31          if (A.difference_of_subtree() > ERROR_BUDGET) then
```

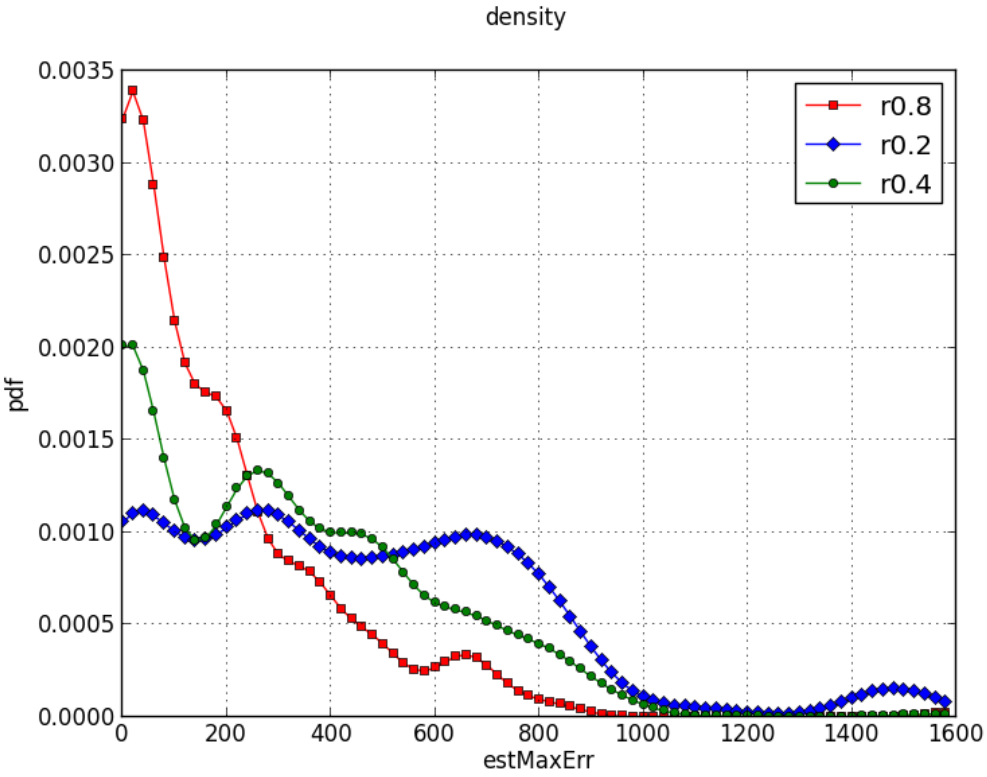

Abbildung 4: Density plot for pattern R2

```
32              Comm.send_to_parent(Neighbors, A.value)
33          end if
34      end while
35  end procedure
```

4. foo

- compare performance for R_1, R_2
  - time series of f (t) and f(t) for r = { 0.2, 0.4, 0.8 } amd $\{R_1, R_2\}$ from the first 5 min
  - trade off plot for R_1, R_2
  - density plot for r = { 0.2, 0.4, 0.8 } and $\{R_1, R_2\}$

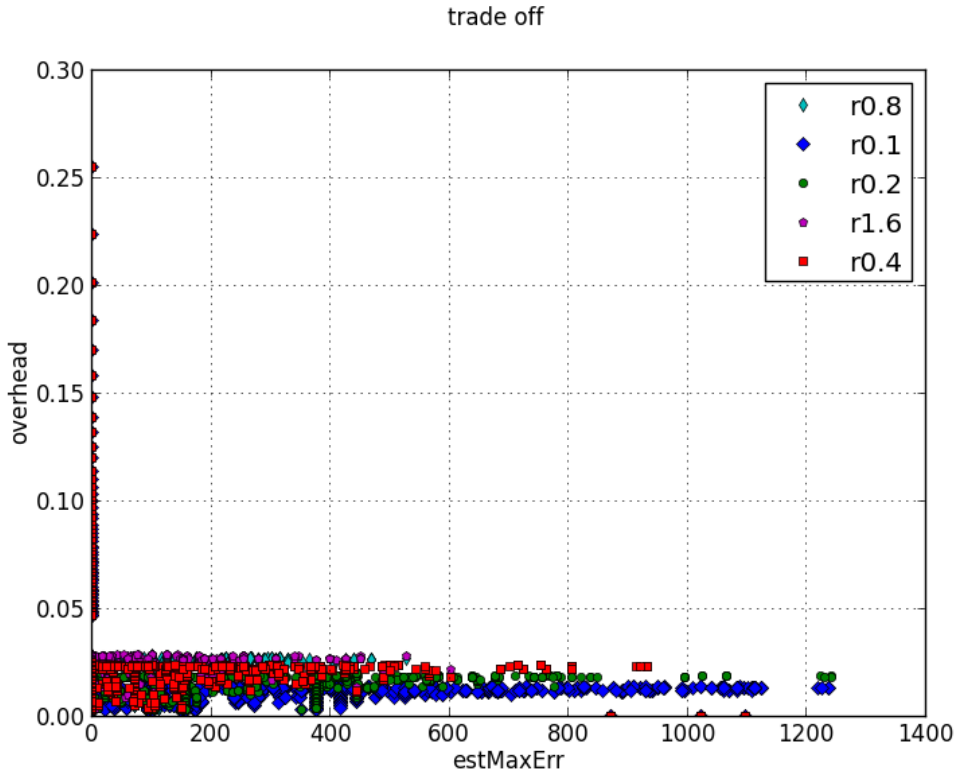

Abbildung 5: Trade-off plot for pattern R1

5. Task II

- pseudo code!
- implementation details
- compare performance for R_1, R_2
  - time series of f (t) and f(t) for r = { 0.2, 0.1, 0.05, 0.025, } amd {R_1, R_2} from the first 5 min
  - trade off plot for R_1, R_2
  - density plot for r = { 0.1, 0.05, 0.025 } and {R_1, R_2}

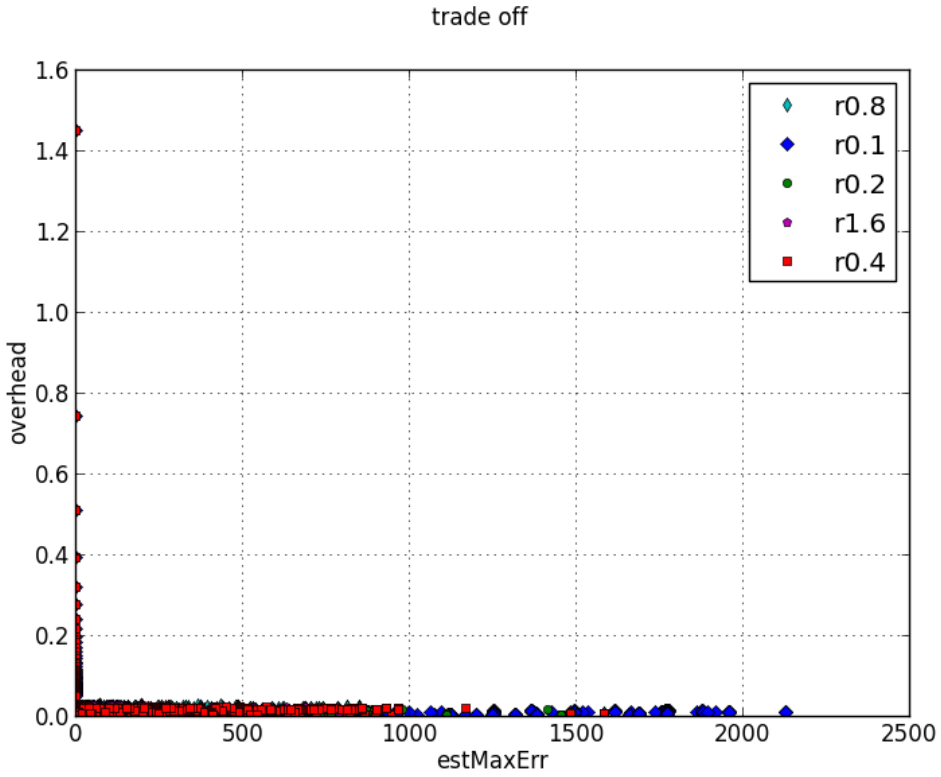

Abbildung 6: Trade-off plot for pattern R2

6. Task III

- pseudo code!
- implementation details
- compare performance for R_1, R_2
  - time series of f (t) and f(t) for r = { 0.2, 0.1, 0.05, 0.025, } amd {R_1, R_2} from the first 5 min
  - trade off plot for R_1, R_2
  - density plot for r = { 0.1, 0.05, 0.025 } and {R_1, R_2}

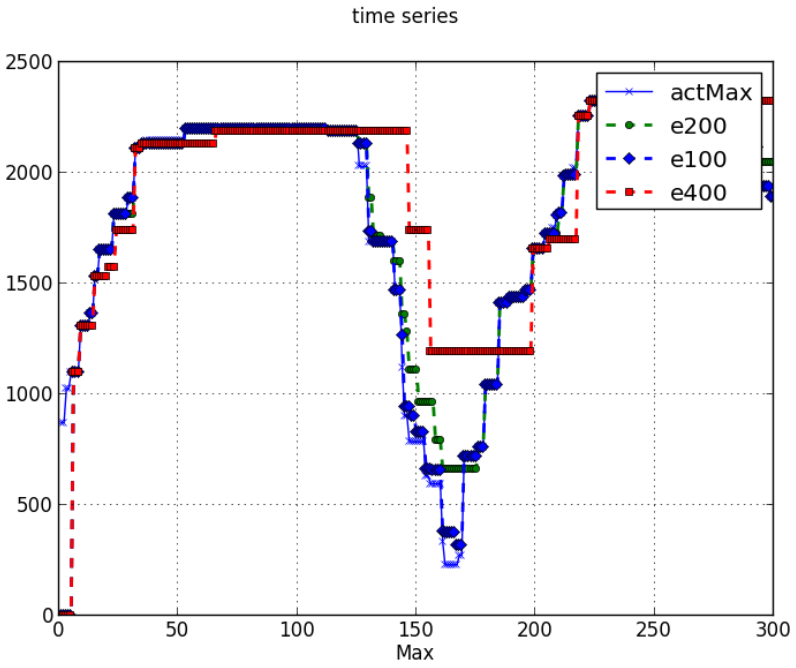

Abbildung 7: Time series for pattern R1

7. Summary

- Compare TaskI, II, III
- Compare R_1, R_2 globally

Literatur

[1] R. Stadler, "Protocols for distributed management," 2012.

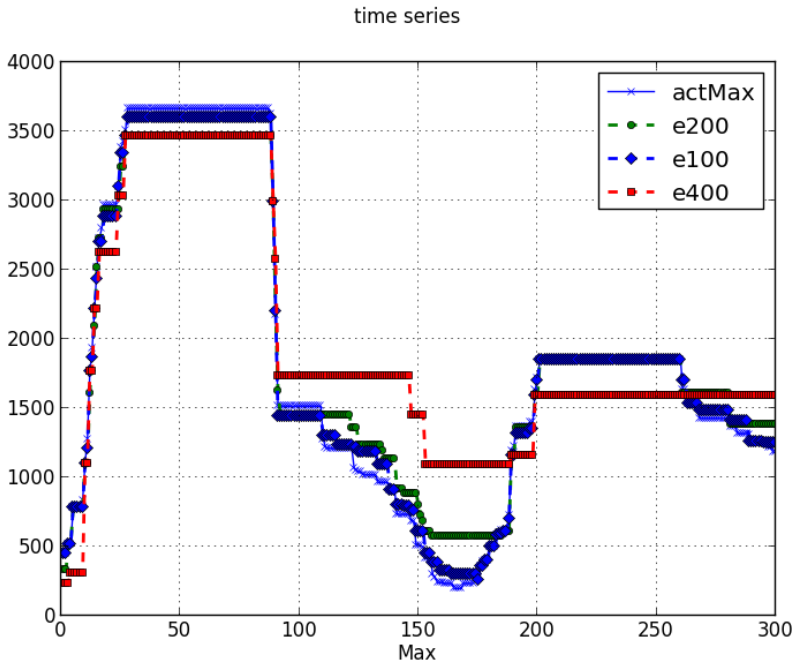

Abbildung 8: Time series for pattern R2

A. Calculations

A.1. Distribution of error budget in aggregation via average

Definitions

ϵ	:= Local error budget of a node	≥ 0	
Φ	:= Overall error budget	≥ 0	
X	:= Previous sum of the response times in subtree	≥ 0	**Aim**
Y	:= Previous number of requests in subtree	≥ 0	
ΔY	:= Number of newly arrived reqeusts in subtree	≥ 0	

It is the aim of this proof to show how the error budget ϵ for every node i can be maximized (ϵ_{max}) while keeping the global error budget Φ constant. The aggregation operation in this case is averaging of the response times of all requests.

Proof

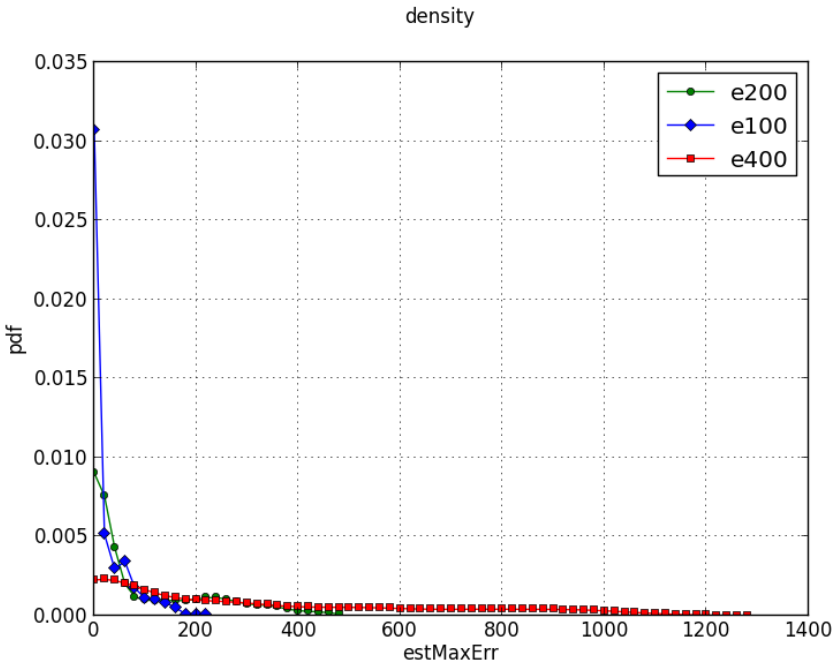

Abbildung 9: Density plot for pattern R1

$$\Phi \geq \left| \frac{X + \Delta Y \cdot \epsilon}{Y + \Delta Y} - \frac{X}{Y} \right| \tag{1}$$

$$\epsilon \leq \begin{cases} \frac{(Y+\Delta Y)\Phi}{\Delta Y} + \frac{X}{Y} & \text{if } Y \neq 0 \\ \Phi & \text{if } Y = 0 \end{cases} \tag{2}$$

The task to find maximum value of ϵ can be transformed to finding minimum value of the right side of inequation 2.

When $Y \neq 0$ we have

$$\frac{Y + \Delta Y}{\Delta Y} > 1, \frac{X}{Y} \geq 0$$

thus

$$\frac{(Y + \Delta Y)\,\Phi}{\Delta Y} + \frac{X}{Y} \geq \Phi$$

When $Y = 0$, the minimum of right side is Φ, thus the minimum value of right side of inequeation is Φ.

It follows $\epsilon_{max} = \Phi$.

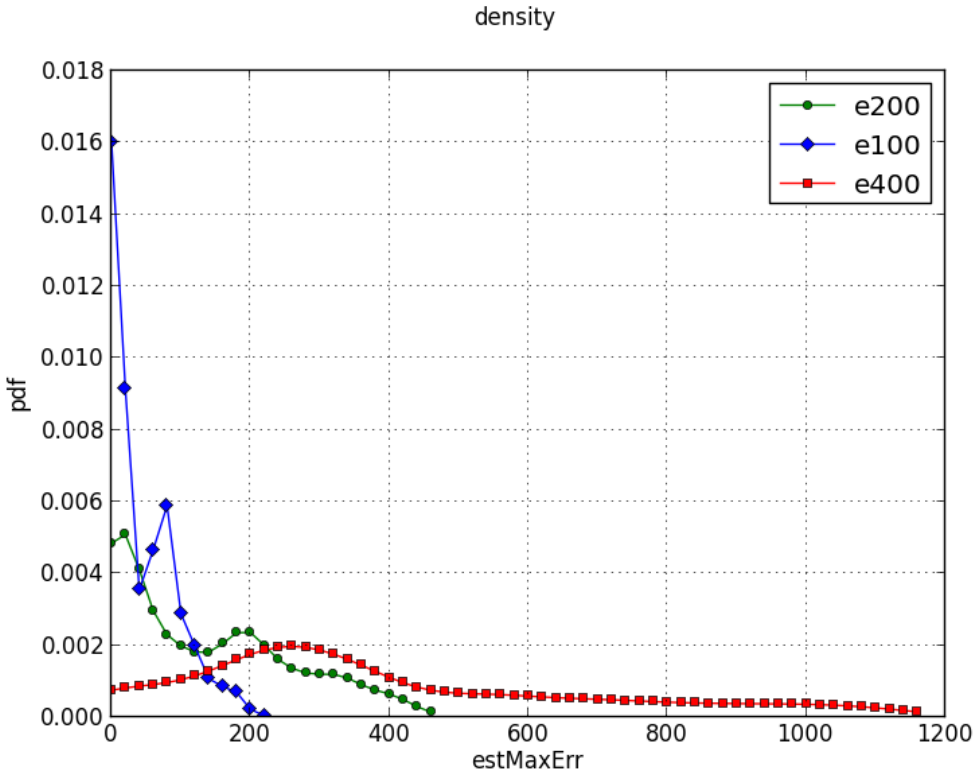

Abbildung 10: Density plot for pattern R2

A.2. Distribution of error budget based on message rate

Definitions

Other than the definitions for problem 3a:

α := Factor of relation between error budget and request rate ≥ 0

δ := Number of newly arrived requests in a node per time unit

Aim

The aim of this proof is to find the maximum factor α relating the request rate δ_i with the the error budget ϵ.

Proof

For every node i with the $S(i)$ denoting all nodes that are direct children of i in the network and i itself.

$$\left| \begin{array}{ll} \Phi \geq & \frac{X+\sum_{j \in S(i)} \epsilon_j \cdot \delta_j}{Y+\sum_{j \in S(i)} \delta_j} - \frac{X}{Y} \\ \epsilon_i = & \alpha \cdot \delta_i \end{array} \right| \tag{3}$$

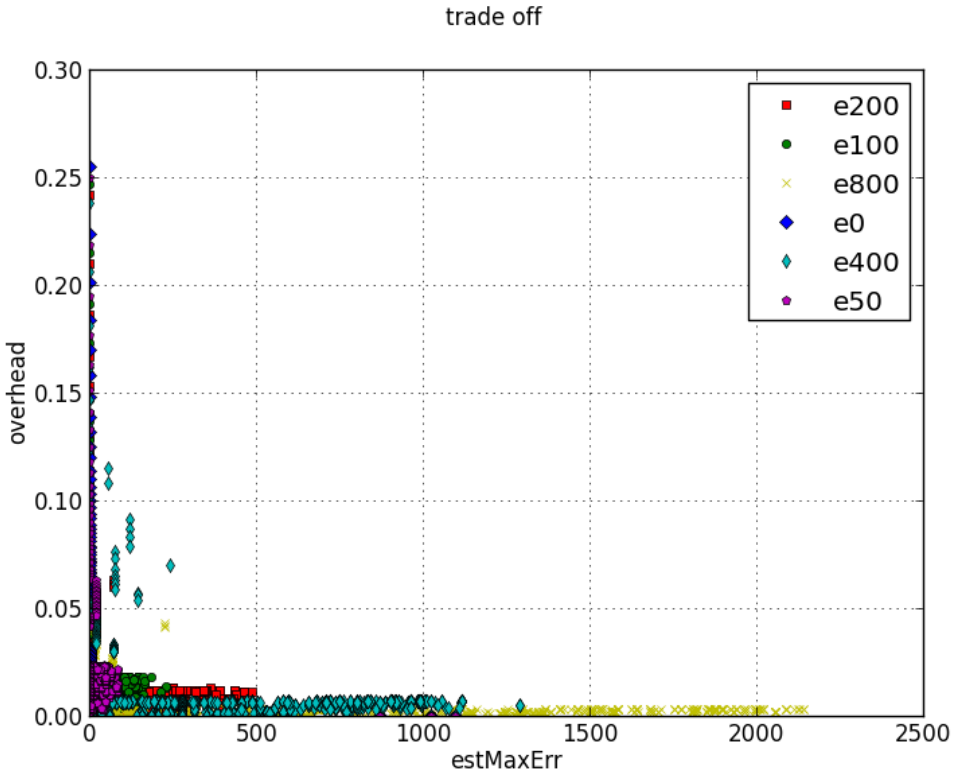

Abbildung 11: Trade-off plot for pattern R1

Applying the same schema of problem 3a, we can have

$$\Phi \geq \frac{\sum_{j \in S(i)} \epsilon_j \cdot \delta_j}{\sum_{j \in S(i)} \delta_j} \tag{4}$$

substitute ϵ_i with $\alpha \cdot \delta_i$, we have:

$$\Phi \geq \frac{\sum_{j \in S(i)} \alpha \cdot \delta_j^2}{\sum_{j \in S(i)} \delta_j} \tag{5}$$

$$\alpha \leq \frac{\sum_{j \in S(i)} \delta_j}{\sum_{j \in S(i)} \delta_j^2} \cdot \Phi \tag{6}$$

Hence the max factor is:

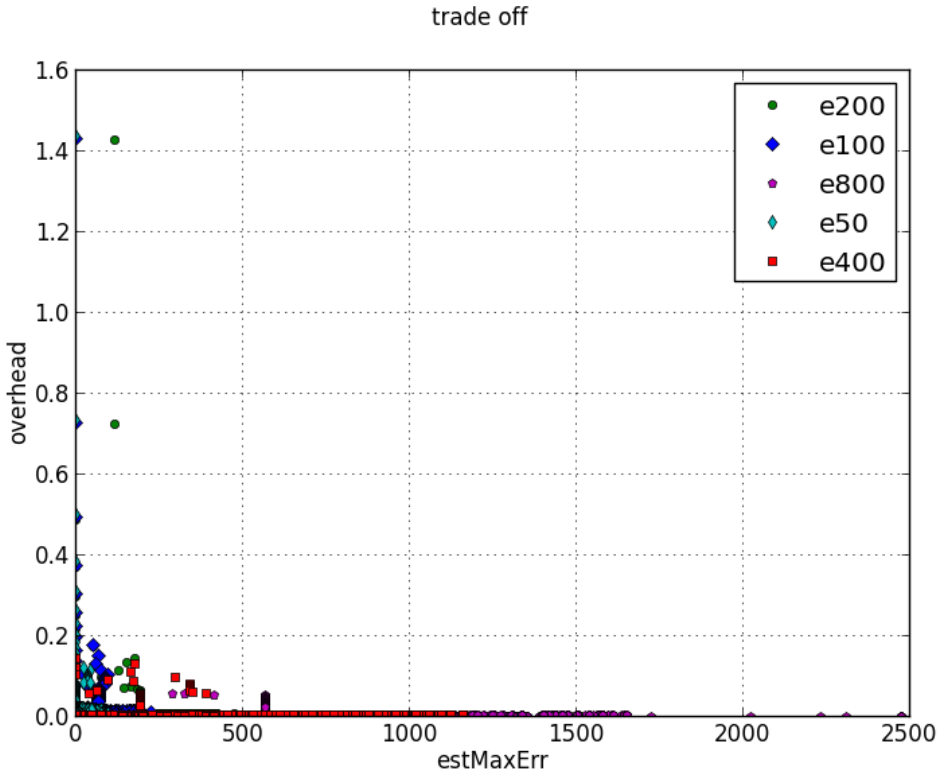

Abbildung 12: Trade-off plot for pattern R2

$$\alpha_{max} = \frac{\sum_{j \in S(i)} \delta_j}{\sum_{j \in S(i)} \delta_j^2} \cdot \Phi \tag{7}$$

Thus error budget for node i is $\alpha_{max} \cdot \delta_i = \delta_i \frac{\sum_{j \in S(i)} \delta_j}{\sum_{j \in S(i)} \delta_j^2} \cdot \Phi$

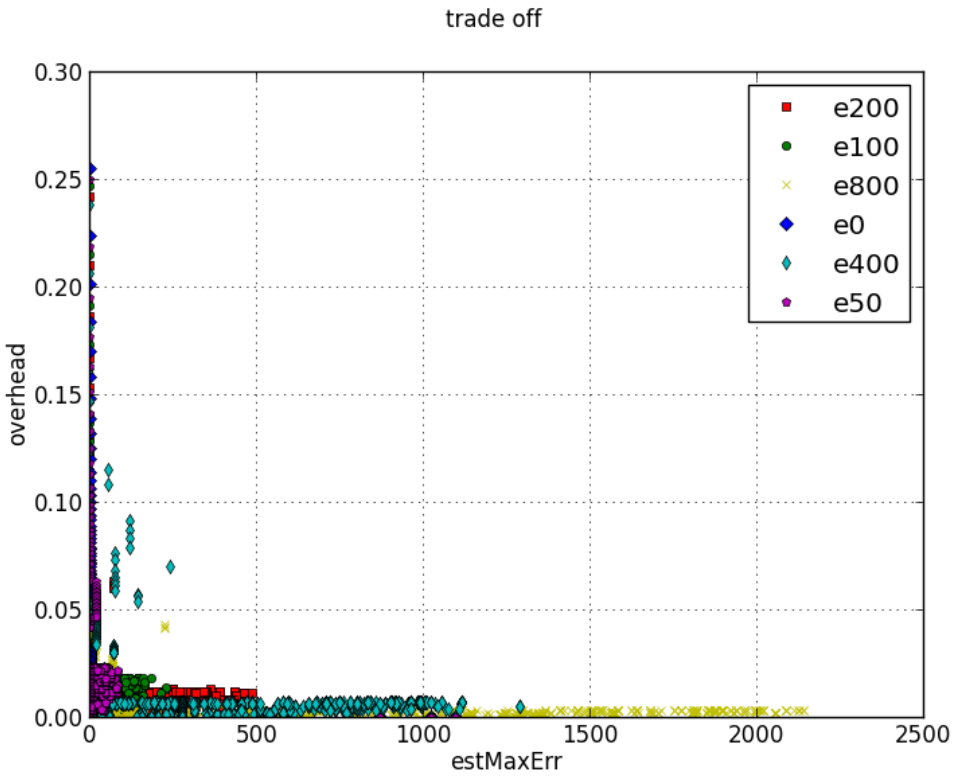

Abbildung 13: Trade off plot

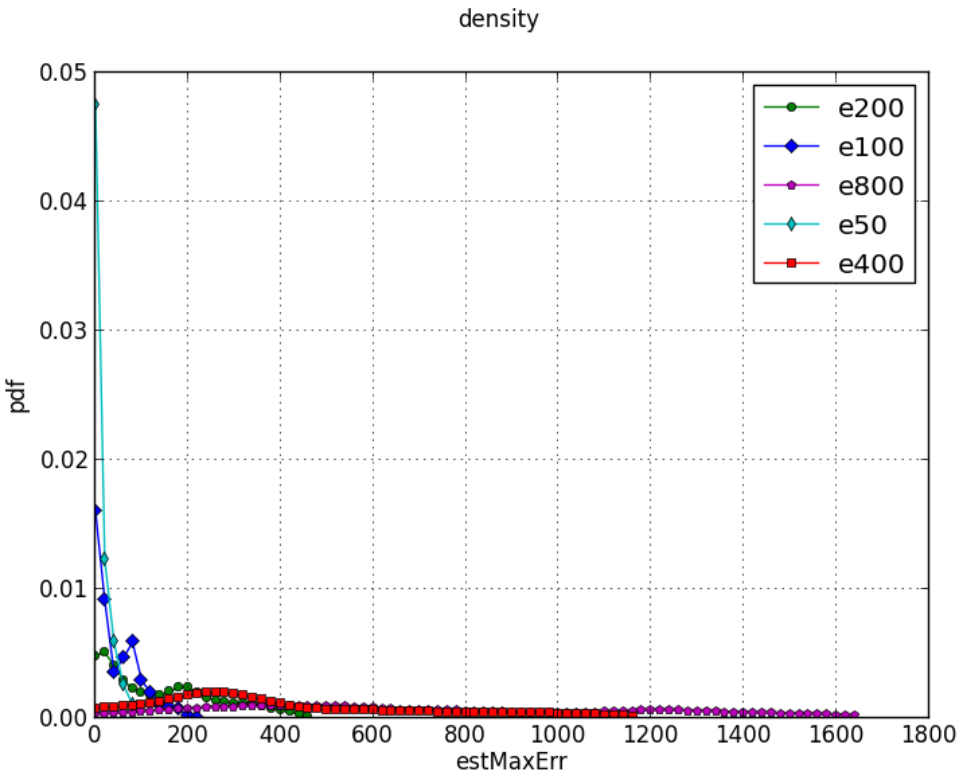

Abbildung 14: PDF plot