

Extremum Feedback with Partial Knowledge

Thomas Fuhrmann¹ and Jörg Widmer²

¹ Institut für Telematik, Universität Karlsruhe (TH), Germany

² Praktische Informatik IV, Universität Mannheim, Germany

Abstract. A scalable feedback mechanism to solicit feedback from a potentially very large group of networked nodes is an important building block for many network protocols. Multicast transport protocols use it for negative acknowledgements and for delay and packet loss determination. Grid computing and peer-to-peer applications can use similar approaches to find nodes that are, at a given moment in time, best suited to serve a request. In sensor networks, such mechanisms allow to report extreme values in a resource efficient way.

In this paper we analyze several extensions to the exponential feedback algorithm [5,6] that provide an optimal way to collect extreme values from a potentially very large group of networked nodes. In contrast to prior work, we focus on how knowledge about the value distribution in the group can be used to optimize the feedback process. We describe the trade-offs that have to be decided upon when using these extensions and provide additional insight into their performance by means of simulation. Furthermore, we briefly illustrate how sample applications can benefit from the proposed mechanisms.

1 Introduction

Feedback is crucial to many networked group applications. The classical example are multicast transport protocols where receivers send negative acknowledgments to initiate retransmissions or report parameters for congestion control [1,7]. Mobile or embedded devices with limited memory need similar mechanisms for flow-control. Feedback is also important for network health monitoring and a variety of distributed computing applications. For example, grid computing applications that employ a huge number of nodes with largely differing capabilities will want to assign each task to the node that is, at a given moment in time, best suited to process the request [2]. A peer-to-peer file-sharing application will want to find the servant that provides the best download capacity and an e-commerce application will seek for the best matching offer at the lowest price.

Although, in principle, feedback could be aggregated by multicast routers, commercial networks do not offer the required functionality. For other networks (e.g., satellite networks) aggregation is not possible at all, since there simply are no intermediate systems. In yet other networks, as for example sensor-networks, responding to feedback requests is very resource consuming, so that it is desirable that only a few nodes need to become active and transmit their responses.

(Note, that in this example we assume also asymmetric costs for reception and transmission of messages. See [4] for a potential usage scenario.)

If aggregation is not possible or inefficient, end-to-end feedback algorithms need to be employed. Such algorithms rely on spreading the feedback across a time interval that allows the group to suppress unfavorable and thus unnecessary feedback responses. The broader this spread (as compared to the network latency) the better the suppression that can be achieved.

In previous work [5,3], it has been shown that exponentially distributed timers are optimal for many feedback scenarios. The algorithms presented there guarantee feedback within a predefined time interval T and avoid feedback implosion, if T is sufficiently large as compared to the network latency τ and an upper bound N to the number of nodes is known. In [6] this exponential feedback algorithm is extended so that extreme values from a group of nodes can be determined.

This paper presents a further extension of the exponential feedback mechanism, namely a mechanism exploiting the knowledge nodes have about the distribution of the values that are to be reported. Such knowledge can typically be expected for most of the practical usage scenarios. In particular, we study cases where the values are *independent and identically distributed to one of several possible distributions*. Not knowing which distribution applies in a given case makes this problem hard. Again, this can be considered a rather typical case. Network measurements indicate that loss characteristics and round-trip-times follow known distributions, but the parameters of the distribution can quickly change when the network becomes congested. Sensors reporting physical values (e.g., temperature or humidity of the environment, etc.) often reflect a common distribution whose parameters change simultaneously for all sensors. Requests in file-sharing peer-to-peer networks show largely differing popularity, where a few requests can be served by a large number of nodes and a large number of requests can only be served by a few nodes. This, again, leads to uncertainty for individual request while good statistical knowledge exists about the distribution of requests in general.

In the remainder of the paper, we will discuss the different feedback mechanisms in Section 2, present further simulation results in Section 3, and conclude with Section 4.

2 Feedback Suppression Mechanisms

In this section, we propose feedback schemes for a potentially very large group of networked nodes to report extreme values to a central instance. We assume that the reported values are received by all other nodes of the group, too. This can be achieved either by using multi-source multicast, with a broadcast medium, or by having the central instance repeat reported values in a suitable manner. We *do not* assume any aggregation capabilities inside the network.

Let $v_i \in [0; 1]$ be the set of values from which we want to know $v_{\min} = \min\{v_i : i = 1, \dots, N\}$. Generalization to other intervals (open and closed) or maximum-

search instead of minimum-search is relatively straightforward by appropriately mapping the respective value interval to $[0; 1]$. From prior work [5,3] it is known that the following algorithm is optimal for the binary case $v_i \in \{0; 1\}$ (e.g., to report packet loss ($v_i = 0$) in a reliable multicast transport protocol).

Algorithm 1 (Binary feedback) *Each node with $v_i = 0$ draws a uniformly distributed random variable $x_i \in [0; 1]$ and sends a report at $t_i = T \cdot \max\{0; 1 + \log_N x_i\}$ unless it knows that another node has answered prior to that time.*

If no feedback is received from the receivers within the time interval T then $v_{\min} = 1$. Here, T is the maximum feedback delay and N is an upper bound on the number of responders.

Extending this algorithm, [6] describes a mechanism to produce v_{\min} for general distributions of $v_i \in [0; 1]$. To this end, it makes use of that fact that a random variable can often be described by a probability density function $p(v)$. If the v_i are independent and identically distributed (iid), the following algorithm is optimal:

Algorithm 2 (Deterministic feedback) *Each node i calculates $x_i = P(v_i)$ and sends a report at $t_i = T \cdot \max\{0; 1 + \log_N x_i\}$ unless it knows that another node has already reported its value.*

Here, $P(v) = \int_0^v p(v') dv'$ is the cumulated probability distribution function. In other words, the random variable x employed in Algorithm 1 is substituted by the random variable v . This is achieved by using a mapping that turns the distribution of the values into a uniform distribution, namely the mapping $P : [0; 1] \rightarrow [0; 1]$. As a consequence, minimal values are reported early, thereby increasing the probability for suppressing unnecessary reports of non-minimal values.

Algorithm 2 is optimal for the case of an iid random variable with known probability density function $p(v)$. In practice, however, the values that are to be reported are rarely iid to *one* known probability density function. Therefore, in the following sections more complicated but also more realistic scenarios are analyzed and respective solutions are derived.

In order to judge the effectiveness of an algorithm, it is important to know the expected number of responses. Ideally, the fraction of nodes that respond at (or before) a given moment should rise exponentially over time. As described in [3] this accounts best for an unknown group size. Conversely, considering ever shorter time-intervals the fraction of nodes responding in any of these intervals should remain constant. Hence, plotting the fraction of responding nodes as area below a *response density* function on a (reversely) logarithmic axis, the ideal curve becomes constant. Note that, according to the fact that there is a finite probability for a node to not delay its answer, the origin of the time axis *does not* lie at the leftmost end of the time range. Furthermore, because of the $t = \log_N x$ relationship the response time depends also on the group size estimate N . Therefore, the following graphs are only illustrative and not

exact. Since both, the probability density function and the expected number of responses as a function of time, can be plotted into a square, the resulting overall graph consists of three adjacent squares.

Figure 1 gives an illustration of Algorithm 2. Its single line shows that each value maps to one defined response time. For algorithms containing random elements, the line widens, assigning a whole range of response times to each value. Furthermore, generally, the actual shape of a response time graph does not only depend on the algorithm but also on the probability density functions.

2.1 Semi-deterministic timers with a priori decision

In many cases where the values v_i are not iid to one probability distribution $p(v)$, one can find a set of probability distributions $p_j(v)$ so that with probability π_j the values are iid to $p_j(v)$. In other words, we do not know which probability distribution applies at any given moment in time, but we know that all the values are from the same probability distribution. This is what can typically be expected if the nodes are identically subject to the same causal effects of a random process. In such a scenario we have two different notions of probability:

- π_j is the probability that a distribution p_j applies.
- $P_j(v)$ is the probability of a node i to draw a value $v_i < v$ under the condition that this distribution applies.

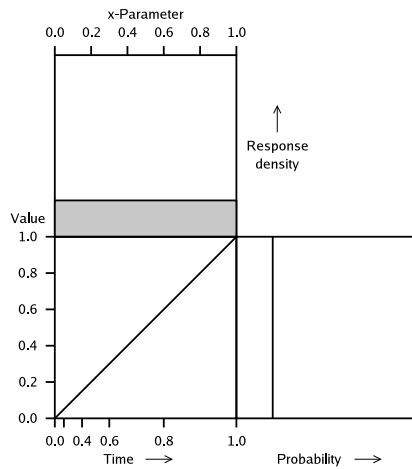


Fig. 1. Uniform distribution of values treated with Algorithm 2

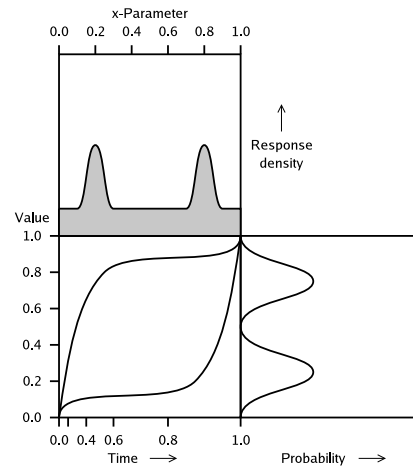


Fig. 2. Response time distribution and expected number of responses for the a priori selection algorithm

There are two obvious algorithms for a priori determination of the response time curves:

Algorithm 3 (A priori mean) *Each node responds at time*

$$t_i = T \cdot \max\{0; 1 + \log_N \sum_j \pi_j P_j(v_i)\}$$

unless it knows that another node has already reported a smaller value.

and

Algorithm 4 (A priori selection) *Each node performs a random experiment such that it responds with probability π_j at time*

$$t_i = T \cdot \max\{0; 1 + \log_N P_j(v_i)\}$$

unless it knows that another node has already reported a smaller value.

Both algorithms are *semi-deterministic* because once the values v_i are known the response times t_i are completely determined. They are *a priori* in a sense that the decision which response time curve applies does not depend on the value v_i .

The problems that can arise from these algorithms are best illustrated by peaked distributions. Consider for example two peaked distributions each of which applies with a probability of 0.5. Using Algorithm 4 results in two significantly differing curves (Figure 2) that are equally populated with responding nodes. The two plateaus in the time domain correspond to the two mean values of the distributions (i.e., each time distribution curve tries to maximally spread out values around the corresponding mean value). Since each curve bears only 50% of the nodes and either of two curves tremendously fails to spread the values in the respective opposite case, 50% of the values are not spread out anymore. The graph shows two peaks in the response density, containing 25% of the nodes each. Only half of the nodes are uniformly distributed over the x -axis.

The mean value algorithm is performing better since it attributes two spreads corresponding to the two peak values (Figure 3). For each case, there is a constant expected number of responses with the expected response time depending on the minimal value within the group. The low value peak corresponds to early responses, the large value peak to late responses. However, compared to spreading the values in both cases over the whole interval, this scenario with a 50% chance each for the low and the high value distribution doubles the respective density of the expected number of responses. Note that the mean value algorithm corresponds to using the combined distribution together with Algorithm 2.

Hence, if the two distributions have greatly differing probabilities (e.g., 90% versus 10%) the distribution with the low a priori probability does not sufficiently contribute to the spread to avoid its values falling into a small time window. This effect thus causes a feedback implosion (see right graph in Figure 3). With a low a priori probability a large amount of responders will start responding almost at the same time. In the example shown above, this will happen at about $0.9T$.

This aptness of feedback implosion is a general and unavoidable feature of all algorithms that are based on a priori selection of a single response time

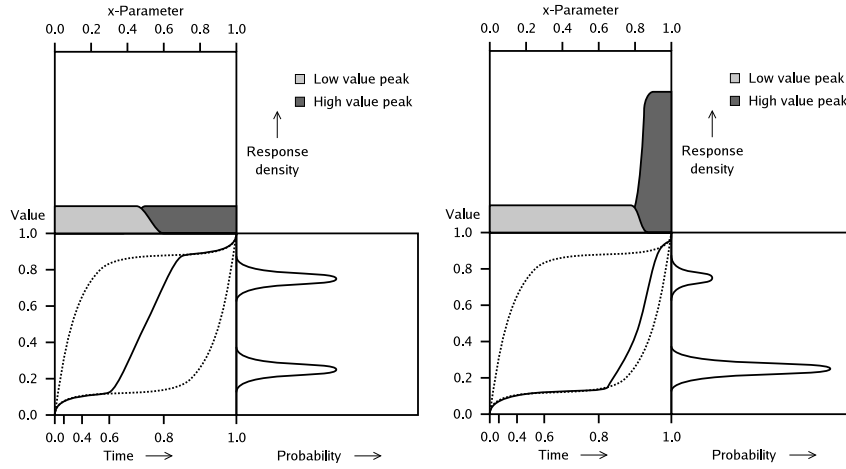


Fig. 3. Response time distribution and expected number of responses for the mean value algorithm. (Left: two peaks with 50% weight each. Right: two peaks with 90% and 10% weight)

curve. Hence, this type of feedback algorithm should not be used in case of asymmetric probabilities for two different probability distributions and in case of many largely differing probability distributions. In fact, it is hard to think of a scenario where the deterministic feedback of Algorithm 2 fails and Algorithms 4 or 3 work well.

2.2 Semi-deterministic timers with value-based decision

One way around this problem is the use of the values v_i not only to calculate the response time t_i but also to decide on the distribution curve that is used for the calculation. As in the previous case, there are two different alternatives: the use of a single mean-value curve and a value-based selection of one response curve among many.

Algorithm 5 (Value-based mean) *Each node i reports its value at time*

$$t_i = T \cdot \max\{0; 1 + \log_N \frac{\sum_j p_j(v_i) P_j(v_i)}{\sum_k p_k(v_i)}\}$$

unless it knows that another node has already reported a smaller value.

The value-based mean value algorithm can produce close to ideal behavior for certain probability distributions while it fails for others. Figure 4 shows three cases each with three peaked distributions of differing width. The response density is exemplarily shown for the case that the real distribution has a large mean value.

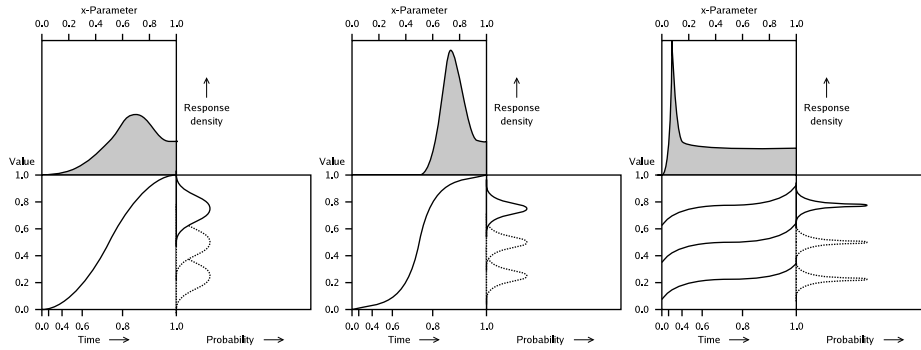


Fig. 4. Response time distribution and expected number of responses for the value-based mean value algorithm with three peaked potential distributions

1. For broad distributions the expected number of responses rises gently to a modest maximum and then falls again to the value that corresponds to an optimal spread. The function mapping values to response times is almost linear.
2. For moderately peaked distributions the rise is much sharper with a large but narrow maximum. Again, towards the limiting value the optimal response density is reached.
3. For narrowly peaked distributions the rise in the response density is very early and very sharp. Almost the whole response interval is covered by an optimal response density. Indeed, the sharp response peak is so small that the probability of a node having a value that falls into this response time tends to zero (i.e., even though the values are closely peaked, they are almost surely optimally spread over the whole response time interval).

At first sight, it might be surprising that the function mapping values to response times is not one-to-one. But a moment's thought shows that this function does not need to rise monotonically: Consider for example three almost completely separated peaks (Figure 4). Then, for each value one can tell with near certainty which of the three peaked distributions applies. Hence, each of the three peaks can be spread over the whole feedback interval, forcing the function to oscillate between early and late response. If, on the other hand, the peaks overlap, many values could either be very small values from a distribution with large mean, or reversely, large values from a distribution with small mean. Both cases would lead to exactly opposite response times. Thus, as a result, the value-based mean value algorithm averages the x -parameter. In case of medium peaks that are too broad to be entirely separated but too narrow to spread the values across a large enough interval, this can lead to a feedback implosion (see Section 3 for an example).

Algorithm 6 (Value-based selection) *Each node i performs a random experiment such that it reports its value with probability*

$$\tilde{\pi}_{ij} = \frac{p_j(v_i)}{\sum_k p_k(v_i)}$$

at time

$$t_i = T \cdot \max\{0; 1 + \log_N P_j(v_i)\}$$

unless it knows that another node has already reported a smaller value.

Algorithm 6 shows the same effect as Algorithm 5 but in a slightly different way. Instead of overshooting at the point where the large value peak becomes more likely than the middle peak, there is a continuing increase in the expected number of responses. As shown in Figure 5, we can observe a considerable risk of feedback implosion towards the end of the feedback interval where suppression is no longer effective.

Thus, although the value-based algorithms are quite capable of spreading responses over the whole feedback interval, there are certain distributions that cause feedback implosion due to peaking or delaying responses: Low value nodes delay their response to allow suppression. If a distribution with a large mean-value applies, both algorithms have to cope with the resulting delay. The mean value version does it as soon as possible and thereby in some cases too fast, causing feedback implosion. The random selection on the other hand adapts more gently but happens to be too slow in some cases.

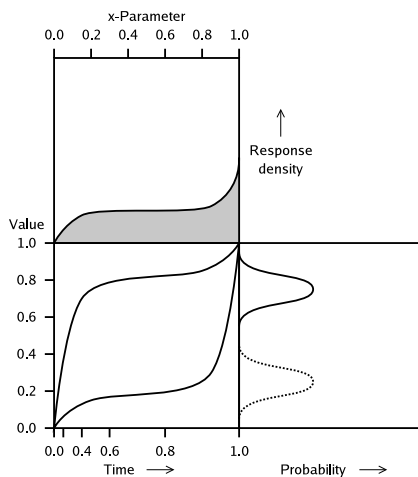


Fig. 5. Response time distribution and expected number of responses for the value-based selection algorithm

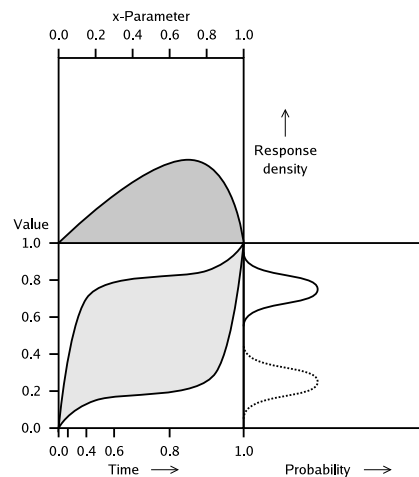


Fig. 6. Response time distribution and expected number of responses for the random spread mechanism

2.3 Randomly Spread Timers

Let $P_L(v)$ and $P_R(v)$ be the upper-left and bottom-right envelope to the cumulated distribution functions $P_j(v)$ in the sense that at least $nP_L(v)$ nodes have a value $v' < v$ and only up to $nP_R(v)$ nodes have a value $v' < v$ where n is a reasonable fraction of the maximum node number N .³

Algorithm 7 (Random spread) *Each node i draws a uniformly distributed random number*

$$x_i \in [P_L(v_i), P_R(v_i)]$$

and reports its value at time

$$t_i = T \cdot \max\{0; 1 + \log_N x_i\}$$

unless it knows that another node has already reported a smaller value.

Like Algorithms 5 and 6, this algorithm is able to spread peaked distributions almost equally over the whole response interval (see Figure 6). Additionally, due to the randomness it is rather insensitive to exceptional value distributions. Even if all receivers report exactly the same value, this algorithm avoids a feedback implosion. Secondly, this algorithm does not require extensive calculations. However, for scenarios in which very little is known about the probability distributions, Algorithm 7 approximates the biased feedback algorithms presented in [6].

3 Simulations

To further illustrate the characteristics of the different mechanisms, we simulate the feedback process with two truncated normal distributions with peaks at 0.2 and 0.8 respectively for the distribution of values at the receivers. Both of the distributions apply with a probability of 0.5. The simulations were carried out with T set to eight times the network latency and for sizes of the receiver set n from 1 to 100,000 receivers. The upper bound on the size of the receiver set N was fixed at 100,000.

As shown in Figure 7, plain exponential feedback suppression achieves the lowest expected number of responses for a given maximum feedback delay T . However, the response times are independent of the response values, leading to an average deviation of the first reported value from the optimum of almost 0.4, when the total number of nodes is close to the estimate N (Figure 8). With deterministic feedback (not shown), the best value is always reported first while maintaining the same average number of responses. As expected, a priori selection and a priori mean frequently lead to a feedback implosion, resulting in an

³ Note that for $n = N$ we would force the envelope to comprise *all* nodes. Choosing $n < N$ prevents outliers from disturbing the overall efficiency.

average number of responses only one order of magnitude lower than the number of nodes. Consequently, these algorithms cannot be recommended for general usage. By using information about the value at the receivers, one would expect to be able to improve the performance of the algorithms. Yet, only the methods of value-based random selection and random spread achieve a reasonable suppression of feedback messages. With a priori selection, feedback implosions may occur at the beginning of the feedback interval, whereas a priori mean and value-based mean lead to a feedback implosions towards the end of the feedback interval.

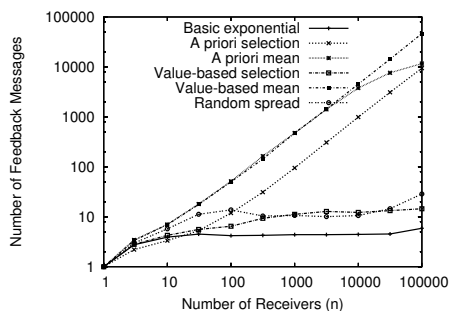


Fig. 7. Number of responses

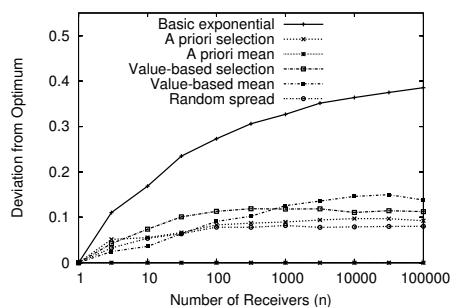


Fig. 8. Quality of the first response

With all proposed mechanisms, the quality of the feedback (i.e., the deviation of the obtained response values from the real optimum of the group) improves significantly. Figure 8 shows the absolute deviation of the value of the first response from the optimal value. As before, we assume the values to be distributed between 0 and 1. A priori mean causes the optimal value or a close to optimal to be reported first almost all of the time. The performance of the other mechanisms is comparable with a deviation of roughly 0.1. Similar results are obtained when analyzing the best of the responses given, instead of the first one. However, such an analysis favors mechanisms which eventually cause a feedback implosions, since the optimal value is likely contained in the large number of responses given.

As mentioned in the previous section, particularly the value-based method depends to a large degree on the underlying distributions. When normal distributions with peaks further apart or closer together are used for the simulations, the method performs much better, as shown in Figure 9 for peaks at 0.1 and 0.9. While still prone to feedback implosion, the average number of feedback messages is reduced by a factor of 50. With value-based selection, which already performed very well in the previous simulation, the number of feedback messages is further reduced to around six messages. As is to be expected, the easier it is to distinguish the two distributions, the better the performance of the mechanisms.

Likewise, the number of responses is significantly better if the two normal distributions have different probabilities. In Figure 10, we depict the average

number of responses when the probability that the distribution with a peak at 0.3 applies is 0.9 and the probability for the distribution with peak at 0.7 is 0.1. The number of responses for the methods where feedback implosion occurs is reduced by a factor of 10 compared to the previous simulations.

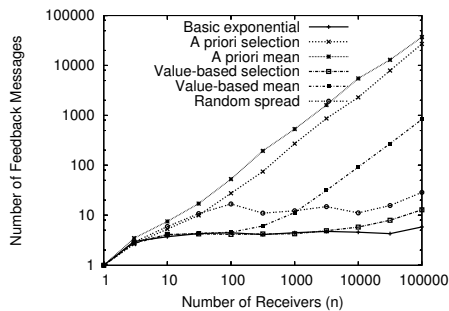


Fig. 9. Responses with peaks at 0.1 and 0.9

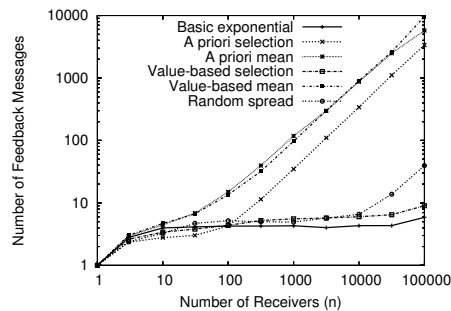


Fig. 10. Responses with distribution probabilities 0.9 and 0.1

Similar simulations with different distributions of response values and different probabilities for these distributions were carried out but have to be omitted for reasons of brevity.

4 Outlook and Conclusion

In this paper, we have analyzed several modifications to the basic exponential feedback mechanism. These modifications incorporate knowledge about the distribution of values that are to be reported into the feedback process. In the simple case of iid values, the distribution function can be used to obtain an optimal feedback algorithm. In more complicated cases, where one out of several potential distributions applies, a trade-off needs to be resolved: If the value distribution is either very broad or very narrow, the *value-based mean value algorithm* leads to a good spread of the responses (see Figure 4). However, for medium peaked distributions this algorithm can cause a severe feedback implosion. The *value-based random selection* and the *random spread* algorithms provide better protection against implosion while falling behind the efficiency of non-random algorithms. Both effects have also been demonstrated by means of simulations.

The proposed mechanisms greatly improve upon the quality of the feedback in case some assumptions about the underlying distribution of the response values can be made. This is often the case when measurement values are to be reported.

In future work, we intend to investigate a number of further issues. So far, the feedback process is independent of possible previous feedback rounds. Since the value distribution often changes on a much larger timescale than the duration of a feedback round, values from previous rounds can be used in addition to

the current value to more reliably infer both, the underlying distribution and the group size. The latter can then be used to improve the efficiency beyond the plain exponential feedback approach. The former extends the applicability of the proposed algorithms to cases where we have no a priori assumptions about underlying distributions, thus leading to a “generic” extremum feedback algorithm.

References

1. Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784 – 803, December 1997.
2. Thomas Fuhrmann, Marcus Schöller, and Martina Zitterbart. Service relocation in programmable networks, 2003. Submitted for publication.
3. Thomas Fuhrmann and Jörg Widmer. On the scaling of feedback algorithms for very large multicast groups. *Special Issue of Computer Communications on Integrating Multicast into the Internet*, 24((5-6)):539–547, March 2001.
4. Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the first ACM international workshop on Wireless sensor networks and applications*, September 2002.
5. Jörg Nonnenmacher and Ernst W. Biersack. Scalable feedback for large groups. *IEEE/ACM Transactions on Networking*, 7(3):375–386, June 1999.
6. Jörg Widmer and Thomas Fuhrmann. Extremum feedback for very large multicast groups. In *Proceedings of Third International Workshop on Networked Group Communication (NGC), London*, November 2001.
7. Jörg Widmer and Mark Handley. Extending equation-based congestion control to multicast applications. In *Proc. ACM SIGCOMM*, pages 275 – 286, San Diego, CA, August 2001.