Real-Time multicast for wireless mesh networks using MPL

Peter van der Stok, 10 May 2014

Version v4

1. Abstract

The performance of the Multicast Protocol for Low power and lossy networks (MPL), as defined in draft-ietf-roll-trickle-mcast, is investigated. MPL is intended for a wireless mesh network. The mesh communication is currently based on the IEEE 802.15.4 standard.

The performance of MPL is expressed in number (percent) of lost messages, average- and maximum end-to-end delays. Performance depends on the forwarder density and number of necessary hops to destination. Performance increase means lowering losses and reducing delay.

To quantify the analysis, simulation is done for a 3x3 mesh network with a one-hop path between every node and for a larger 15x3 multi-hop mesh network needing 4-5 hops for the furthest destination. It is shown that performance can be increased by the addition of a real-time scheduling layer between MPL and MAC.

The document concludes with recommendations for parameter values to minimize delays and losses on the chosen network configurations. It is believed that most of these values also hold for other network topologies, given the underlying configuration independent reasoning.

2. Introduction

Networking of lights in building is necessary for energy saving, such that energy consumption of the lights can be optimized as function of the occupancy of the spaces in the building. Wireless networking is interesting because it reduces cabling costs compared to wired networks. For retrofit of existing buildings, wireless networking avoids re-cabling and other intrusive operations. The wireless devices obtain their power from batteries or from energy scavenging. Currently the IEEEE 802.15.4 standard standardizes a low power wireless interface. On the basis of this standard, it has become feasible to connect lights, sensors, and switches to a wireless network.

A few years ago, most networking standards for building control were driven by manufacturing organizations such as ZigBee, BACnet, and others. The recent standardization of Internet packets over the IEEE 802.15.4 wireless standard makes the wireless IP networking a viable proposition for building networks. Lighting networks have the specific requirement that many lights need to be switched on, off or dimmed almost simultaneously. The Multicast Protocol for Low power and lossy networks (MPL) is currently proposed as an IP standard for that purpose [1].

MPL is derived from the Trickle algorithm [2] that was designed to distribute operational network values to all nodes in the network. New information was distributed within tens to hundreds of seconds and the design particularly tries to minimize the load of the network. MPL is different from Trickle in the sense that the number of repeated packets is limited and packets from different sources are handled independently.

The purpose of this note is to analyse the performance of MPL, and in particular to verify that it scales to the network sizes that can be expected, and that commands arrive within 200ms at all destinations from sending by a source. A simulator [3] is used that executes a detailed model of the IEEE 802.15.4 standard [4]. The results of the simulation suggest clear recommendations for the MPL parameter values to be used during the operation of the network.

The main conclusions are: (1) It is possible to multicast multiple messages to different sets of destinations (e.g. different rooms), (2) within 5 hops multicast messages arrive within 300 ms (3) less than .5 permil of the messages are lost when the network loses 20% of the packets.

It must be noted that simulations of networks help to determine what is impossible, but real life experiments are needed to confirm the performance results of the simulation.

3. Packet propagation

There are two collaborating mechanisms active during the propagation of the packets from source to destination: (1) The IEEEE 802.15.4 MAC that takes care of the point to point transmission of the packet between 2 nodes, and (2) the MPL algorithm that distributes the packets to all destinations by resending received packets to all neighbours. Both algorithms are shortly explained.

3.1.IEEE 802.15.4

For this analysis we use unslotted MAC. Before transmission, a packet can be stored in memory local to the MAC. The packets are served on a first come first served basis. When the memory is full, new packets are rejected by the MAC.

Packets are transmitted following a csma/ca algorithm. Packets are sent consecutively. Before sending a packet, the MAC introduces a delay with a random value in the interval [0, $2^{BE} - 1$] back-off units. BE commonly has a value between 1 and 5 and a back-off unit takes .32 ms. After the delay the MAC senses if the medium is free. When the medium is free, the packet is sent. When the medium is occupied, again the random delay is done with an increased BE value up till a maximum value, followed by the sensing of the medium. This is repeated *macMaxCSMABack-off* times. When after the retries the packet could not be sent, the MAC rejects the packet.

3.2.MPL algorithm

The MPL algorithm is described and motivated in [2]. MPL specifies two execution modes: Reactive and proActive. The analysis in this paper is limited to the proActive mode. MPL forwarders accept multicast packets after the enabling of the interface with the multicast address. A *seed* generates and sends multicast packets which are consecutively numbered. At reception of a new packet, the MPL forwarder starts a series of consecutive *trickle timer intervals*, where the first interval has a minimum size of I_{min}. Each consecutive interval is twice as long as the former with a maximum value of I_{max}. There is a maximum number of intervals given by *max_expiration*. For each interval of length I, a time *t* is randomly chosen in the period [I/2, I]. For a given packet, p, during an interval of length I, MPL counts the number of times it receives p during the period [0, t] in a counter c. At time t, MPL re-broadcasts p when $c < k$, where k is a predefined constant with a value $k > 0$.

The packet sent by the seed is called packet S0. Packets sent in consecutive trickle timer intervals are called packets Sx with x= 1, 2, according to the trickle timer interval instance.

4. Delay sources

When a seed sends a packet, p, there are several contributions to the time it takes to transport p from the seed to a given destination, D. The history of p is reflected in Figure 1.

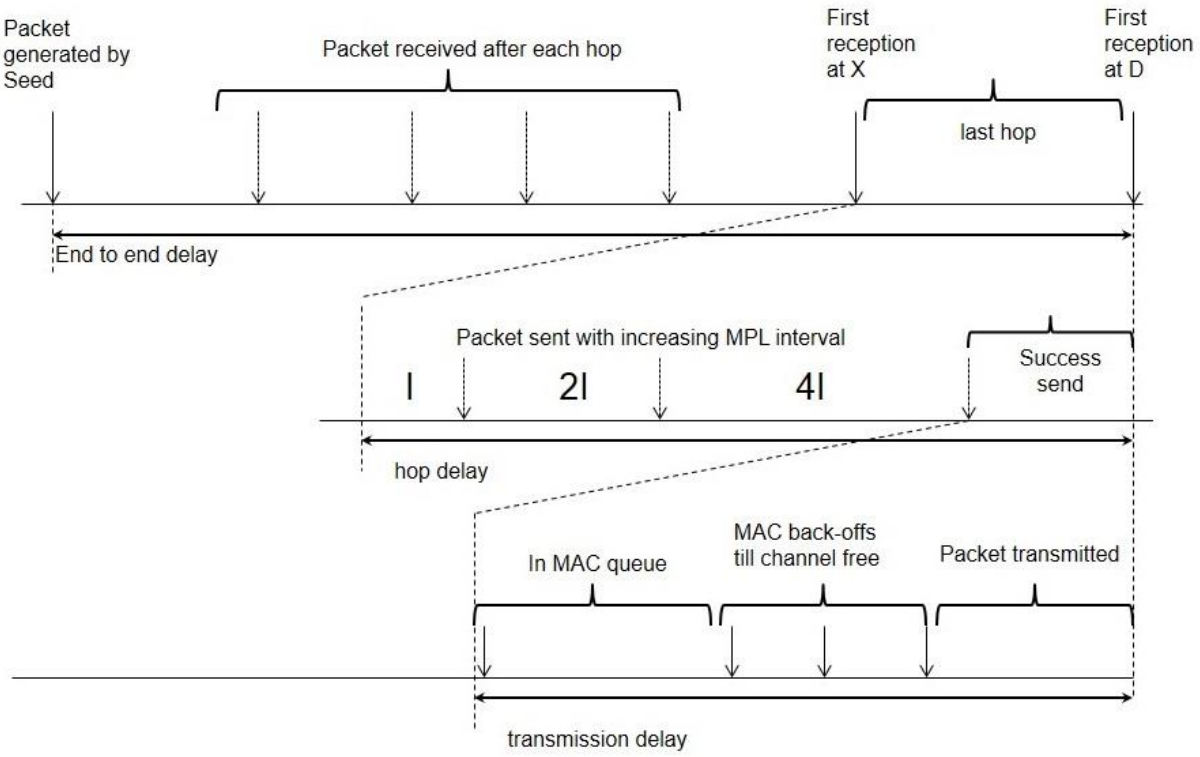

Figure 1, propagation history of packet

The packet is generated and sent by a *Seed*, received and sent on by a sequence of zero or more MPL forwarders, and finally transmitted from the last forwarder *X*, in one hop to the destination, *D*. The *end-to-end delay* of packet, p, is defined by the moment p is generated by the seed till the first reception at D. The *hop delay* is defined by the moment X receives p till the moment of reception by D. The *transmission delay* is defined by the moment X inserts p into the MAC till the moment of reception by D. There is a difference in packet handling between the seed and each forwarder.

- The seed sends the packet immediately after its generation and starts the MPL protocol if it is an MPL forwarder; consecutively, the packet is re-sent during the first and following trickle timer intervals.
- After each hop, arriving at a forwarder, the packet is NOT sent immediately after reception but during the first and following trickle timer intervals.

In a large majority of use cases, the packet is sent directly from the seed to a destination in one hop. When the packet is directly received by D after the first transmission by the seed, the three delays are equal.

5. MPL timing behaviour

MPL is proposed for switching on lights with the requirement that end-to-end delays are as low as 200 ms. In some cases the seed is as far as 4-5 hops removed from the final destination. Expressions are found for the transmission delay, the hop delay and the end-to-end delay in the text below.

5.1. Transmission delay

The transmission delay is determined by several parameters manipulated in the simulation:

- The size of the queue of the MAC
- The number of back-offs and the minimum and maximum BE values (see section 3.1)
- The time it takes to send a packet

The default values for the MAC is min BE = 3 and max BE = 5 with the number of back-offs equal to 3. The delay from the first back-off delay to "sensing free" can fluctuate between 0 and $(2^5+2^4+2^3)$ times .32 ms, which takes 0 - 18 ms. The time for sending a packet is 3.4 ms. Under load, the transmission delay takes a minimal 3.4 ms and a maximum of 21.4 ms. When packets are stored in the MAC queue, assuming a queue of 2 packets, the transmission delay from insertion into the MAC to final arrival can be as large as 3*(18 +3.4) ms = 64.2 ms. Transmission delay (TD) lies between: 3.4 < TD < 64.2.

5.2. Hop delay

For this section we consider Max_expiration to be equal to 2. Later it is shown that this is a reasonable choice. Figure 2 shows the hop delay for the sending and repeating by a seed with inclusion of the possible waiting with trickle timer intervals starting with I_{min} = 10 ms. Packet S0 is the original packet sent by the seed. Packets X1 and X2 are sent by forwarder X, or by a seed, S, with a forwarder function sending packets S1 and S2.

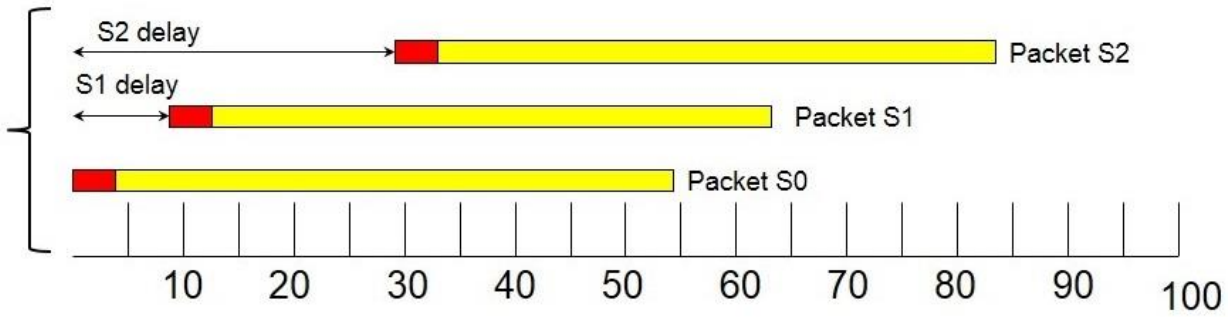

Figure 2, Hop delay from seed

Figure 2 shows bars with a first red part of 3.4 ms representing the minimum time needed to transmit a packet. The yellow part represents the interval that can be occupied by waiting in the MAC as calculated in section 5.1. It can be seen that the MAC back-offs take the largest part in the hop delay. The transmission intervals of the repeated packets are represented in parallel. This is virtually impossible with I_{min} < back-off wait, because packet S1 is stored in the MAC while packet S0 is still back-offed. The worst case hop delay occurs when packets S0 and S1 are lost, followed by packet S2 sent simultaneously with 2 other packets, leading to: 3.4 <Hop delay < 3*I_{min} + TD.

In a simple forwarder X (not seed) packet X0 does not exist. The maximum hop delay of a forwarder occurs when packet X1 sent at I_{min}/2, leading to: I_{min}/2 + TD < hop delay < 3*I_{min} + TD..

5.3. End–to-end delay

The end-to-end (e2e) delay is dominated by the I_{min} value of MPL when back-off in the MAC is ignored. After arrival at a forwarder the packet X1 is repeated at time t in the interval [I_{min}/2, I_{min}]. In the best case, a packet is repeated in the first interval after arrival, and is not delayed in the MAC. Assuming n hops, the best case end-to-end delay, with I_{min}/2 < t < I_{min}, lies between:

$$3.4 + (n-1)*(3.4 + I_{min}/2) < \text{e2e delay} < TD + (n-1)*(TD + I_{min})$$

Where 3.4 ms is the interval needed to transmit a packet by an interface. The worst case transmission happens when packet X1 is lost, and packet X2 is sent at the end of the second trickle timer interval. Replacing I_{min} by 3*I_{min}, and I_{min}/2 by 2*I_{min}, the worst case end-to-end delayl of X2, assuming that the MAC does not back-off, lies between:

$$3.4 + (n-1)*(3.4 + 2*I_{min}) < \text{e2e delay} < 3.4 + (n-1)*(3.4 + 3*I_{min})$$

6. MPL parameter value relations

6.1. I_{min} and k

The k-parameter in Trickle, taken over by MPL, is quite important to make sure that the number of repeated packets remains bounded. The Max_expiration parameter of MPL, not present in Trickle, significantly assists in reducing packet load. In a control environment, emphasis is placed on the timely distribution of sensor values. The wish is to quickly disseminate a message to all receivers by sending the packet in the first repeat interval. Too low values of k can lead to significant packet removal at the first repeat, at the detriment of MPL performance. This section shows how much the value of c is increased with high probability for the first and second repeat packets, thus indicating a minimum k value.

For example, Figure 3, Figure 4, and Figure 5 show that a message is broadcast from a seed and starts up the trickle timer intervals of forwarders x and y after *delay*, the time needed for the transmission of a packet. Forwarder x is the forwarder which sends the first packet out of n forwarders which receive a packet from a given seed. Forwarder y stands for 0 or n-1 forwarders which received the same packet from the seed as forwarder x. The probability that one of the n forwarders is forwarder x is in general 1/n. Assuming that in the considered deployments all n forwarders can reach the same set of nodes, it is sufficient to analyse the behaviour of forwarder x.

In interval ($I_{min}/2$, I_{min}), the seed repeats the message, which arrives after delay at forwarder X. The following 3 cases are discussed to calculate the probability that c is increased in forwarder x: (1) I_{min} < delay, (2) $I_{min}/2$ < delay < I_{min}, and (3) delay < $I_{min}/2$. Given that delay = 3.4 ms, the 3 conditions can be formulated as (1) I_{min} < 3.4 ms, (2) 3.4 ms < I_{min} < 6.8 ms, and I_{min} > 6.8 ms.

The following terminology is introduced:

- Si is the i^{th} packet sent by the seed, with i= 0, 1, or 2.
- Xi is i^{th} packet repeated by forwarder X with i =1 or 2.

Case 1 *(I_{min} < delay)*: Figure 3 (left) shows that the seed sends S0 at time=0. The seed sends S1 at time t, when the original message, S0, is still in transit and S1 cannot be sent, because the medium is occupied. After at least delay time units S1 is sent by the seed. At time tx with delay+$I_{min}/2$ < tx < delay +I_{min}., X1 is submitted to the MAC. Packet S1 will arrive after 2*delay at forwarder x which is larger than delay+I_{min} > tx. A forwarder y can send Y1 after the transmission of S1 because S1 occupies the medium. Consequently, c is never increased in forwarder x for the first repeat message, X1.

Consider packet X2 sent by forwarder x. Figure 3 (right) shows that S1 arrives at 2*delay > I_{min}+delay at forwarder X. Consequently, packet X1, submitted to MAC at t_1, is sent at time 2*delay. The next packet (S2, or Y1,..) arrives at 4*delay > 3*I_{min}+delay. Packet S1 increases c for packet X2, and no other packet can arrive before 3*I_{min}+delay. When S2 is at the start of the interval (2*I_{min}, 3*I_{min}), S2 can be sent before X1 with a probability 1/(n+1), and c may be incremented twice for packet X2.

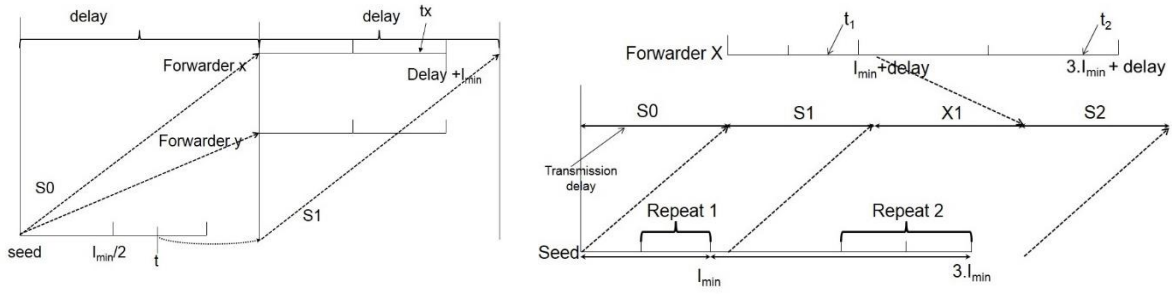

Figure 3, arrival of repeated message from seed to forwarder, for I_{min} < delay; left X1, right X2

Case 2 *($I_{min}/2$ < delay < I_{min})*: Figure 4 (left) shows that the message, S1, submitted to the MAC by the seed at time t < delay, is sent at time t_E = delay, and arrives at forwarder x at time 2*delay < I_{min} + delay. Assume the X1 packet is submitted to the MAC at time tx. The S1 packet of the seed can increase c when tx >

2*delay. The value of c of packet X1 cannot be increased beyond 1, because the packets repeated by forwarders y arrive after an interval of 3*delay > I_{min}+delay, because Y1 is sent after X1 by definition.

The probability that c is incremented to 1 for packet X1 can be calculated. Set probability P = 2*(I_{min} – delay) /I_{min}. The value of c is incremented for X1 under 2 conditions: (1) when S1 is submitted to the MAC before t_E, and all n forwarders submit Y1 in the interval (2*delay, I_{min} + delay), and (2) packet S1is submitted after t_E, and all forwarders submit Y1 in interval (2*delay, I_{min} + delay) such that S1 arrives before Y1 is submitted to the MAC. Packet S1 arrives before t_E with probability 1-P and after t_E with probability P. The n forwarders submit in interval (2*delay, I_{min} + delay) with probability P^n. The probability that S1 arrives before all forwarders submit in interval (2*delay, I_{min} + delay) is given by: 1/(n+1). The probability P1 that c is incremented to 1 for packet X1 is given by: P1 = (1-P)P^n + P^{n+1}/(n+1). P1=0 for I_{min}=delay, and P1=1/(n+1) for I_{min}/2 = delay.

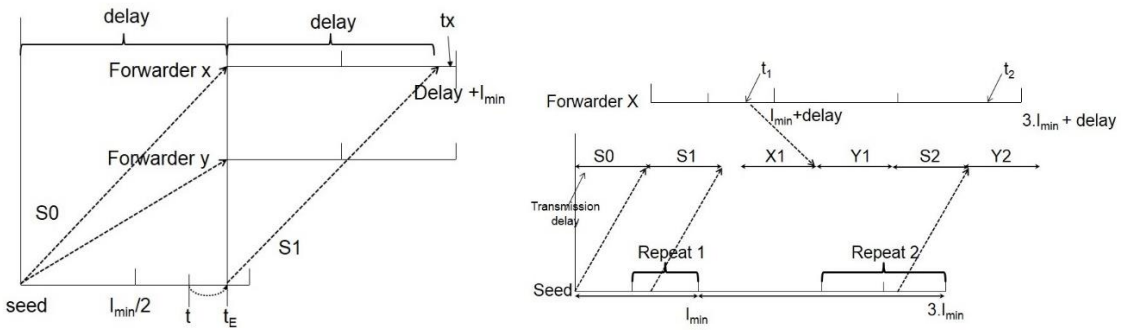

Figure 4, arrival of repeated message from seed to forwarder, for I_{min}/2 < delay < I_{min}; left X1, right X2

Figure 4 (right) shows that after the arrival of packets S1 and S2, packet X1 can be submitted to the MAC at time t_1. Submission to MAC at time t_1 can occur before arrival of S1. Packet X1 is then sent immediately after the arrival of Packet S1. Incrementing c for X2 starts after the arrival of X1. The maximum number of packets to arrive after X1 and before sending X2 takes place when I_{min}/2 = delay. Up to three packets (for example Y1, Z1, and S2) can increase c for packet X2 of forwarder X. The maximum is minimum (3, n) with n the number of forwarders.

Case 3 (delay < I_{min}/2): In Figure 5 (left), Seed submits packet S1 at time t_1 in interval [I_{min}/2, I_{min}], which arrives at forwarders y in interval [delay+I_{min}/2, delay+I_{min}]. Forwarders y submit packets Y1 in interval [delay+I_{min}/2, delay+I_{min}]. The value of c for Y1 is incremented to 1 when the packet S1 arrives before the forwarder submits Y1. The value of c for X1 is not incremented when S1 arrives after the submission of X1 by the first forwarder x. Consequently for X1, c = 1 with probability 1/(n+1) and c = 0 with probability n/(n+1).

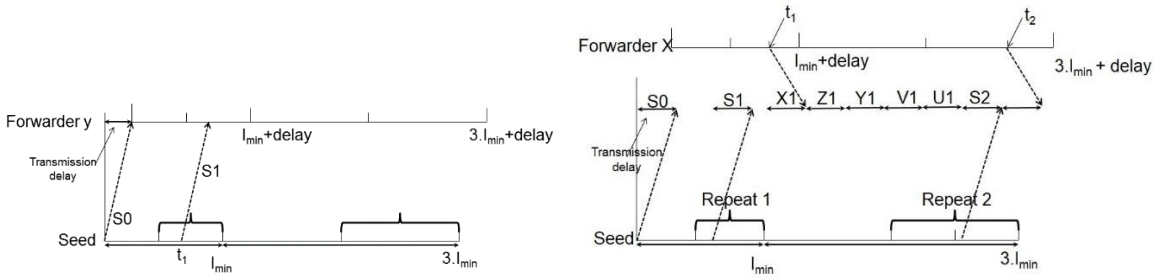

Figure 5, Probability that c is increased for delay < I_{min} /2; left X1, right X2

Figure 5 (right) shows an example where c of X2 is incremented to 5. Assume that apart from forwarder X, also forwarders U, V, Z receive packet S0 from the seed. The seed sends S0 followed by S1. At time t_1, forwarder X is the first forwarder to send packet X1. Every time the medium is liberated, the following packet is sent, shown by the sequence Z1, Y1, V1, U1, concluded by S2. All of them arrive at forwarder X before time t_2 at which X2 is submitted to the MAC by forwarder X. In this example the counter c for X2 is incremented to 5. The length of the sequence of incrementing packets is limited by the number of packets arriving at forwarder x in interval [I_{min}+delay, t_2], and the number of forwarders Y sending a packet Y1. When delay is sufficiently small, c is incremented to n with n*delay < 2*I_{min}. The maximum increment of c for other forwarders than x can increase to 2n-1.

6.2. k-settings

In

	I_{min} < delay	$I_{min}/2$ < delay < I_{min}	delay< $I_{min}/2$	
C max	0	1	1	X_1
probability	1	0< P1 <1/(n+1)	1/(n+1)	
C max	0	1	$\lceil I_{min}/2delay \rceil$	Z_1
	1	0< P1 <1/(n+1)	Not calculated	

Table 1 the maximum possible c values and their probability of occurrence are shown for packets X1, Z1 sent by forwarders X, Z, where X is first and Z last forwarder to forward S0. The minimal value k = 2 is required to assure that c<k, and packet X1 is sent. Choosing k=2 implies that only forwarder X repeats X1. Given that packets are lost, choosing k>2 is recommended.

	I_{min} < delay	$I_{min}/2$ < delay < I_{min}	delay< $I_{min}/2$	
C max	0	1	1	X_1
probability	1	0< P1 <1/(n+1)	1/(n+1)	
C max	0	1	$\lceil I_{min}/2delay \rceil$	Z_1

	1	0< P1 <1/(n+1)	Not calculated	

Table 1, probability of maximum c values for different I_{min} and delay values

6.3. Buffer space

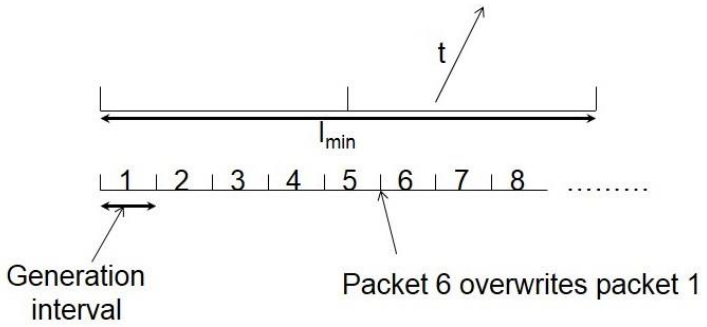

Figure 6, Overwriting of packet information

MPL maintains circular buffers of packets sent by a seed. The information for a given packet is necessary to repeat a packet. Figure 6 shows that packet information is overwritten when the circular buffer is full and new packets arrive. In each forwarder, MPL maintains a buffer for each seed of recently received packets that need to be transmitted. In the lower part of Figure 6 the regular insertion of a packet into the buffer is shown as determined by the generation interval of the seed. In the figure, given a buffer space of 5 packets, packet 1 is overwritten by packet 6, packet 2 by packet 7, etc. At the time packet 1 is received the trickle timer interval is started and the packet is resent at time t, with $I_{min}/2 < t < I_{min}$. The resending only takes place when the packet is still in the buffer, which imposes that t < 6*T_generation. This shows the importance of allocating a circular buffer size that is sufficiently large. The necessary buffer space is given by:

$$\text{Nr of packet buffers} > (2^{max\_expiration}-1).I_{min}/T\_generation$$

7. Real_Time scheduling

Given the large contribution of the MAC back-off to the maximum end-to-end delay (see section 5), a different approach is tried out by re-sending or rejecting a packet as function of its deadline or sequence number. Because MPL specifies that a packet is re-sent from the network layer, resending from the MAC is not really needed, and the accompanying back-off delays can be suppressed almost completely. To support real-time scheduling at the network layer, the MAC parameters are set such that minBE is equal to maxBE is equal to one, and the maximum number of back-offs is set to 1. The consequence is that the maximum sojourn time of a packet in the MAC is limited to 3.7 ms.

7.1. Scheduling algorithm

When the MAC is busy, packets can arrive to be inserted in the MAC. Given that no packets can be stored in the MAC, a 1-place buffer is introduced for each seed to temporarily store newly arriving packets. Maximally two packets are active during transmission: (1) one offered to the 1-place buffer, and (2) one stored in a 1-place buffer by MPL.

A deadline can be allocated to each packet. The deadline associated with a packet is used to remove a packet when it is too old and cannot satisfy its timeliness requirements. The value of the deadline is taken with respect to the generation time of the packet by the seed. The deadline can be checked against expiration in the following way:

- With synchronized clocks, the deadline is 200 ms + generation time.
- Without synchronized clocks, the deadline is the actual clock time + 200 – 20*hops.

Without synchronized clocks it is assumed that each hop takes about 20 ms. Other numbers can be chosen dependent on requirements and network performance. The first approach necessitates transport of the generation time or deadline, and the second approach necessitates transport of the number of hops.

Given multiple seeds, a 1-place buffer per seed is present.

There are two events at which the contents of the 1-place buffer or the processing by the MAC can be changed. (1) MPL submits a packets to transmit, or (2) the MAC returns a packet, either sent or not sent. When a packet arrives from MAC or MPL, it is stored in a working buffer. Call the packet in the working buffer the WBP and the packet in the 1-place buffer the 1BP. When the working buffer and the 1-place buffer are filled, a choice presents itself about which packet to store into the 1-place buffer. The scheduling algorithm RT_i implements the choice.

Assuming that the deadline of the packet is larger than the actual clock time, MPL inserts the packet in the working buffer, and the following actions are undertaken (see Figure 7):

- When the MAC is free, the WBP is submitted to the MAC.
- When the MAC is busy and the 1-place buffer is empty, the WBP is stored in the 1-place buffer
- When the MAC is busy and the 1-place buffer is filled, **choice A** is solved by the RT_i algorithm.

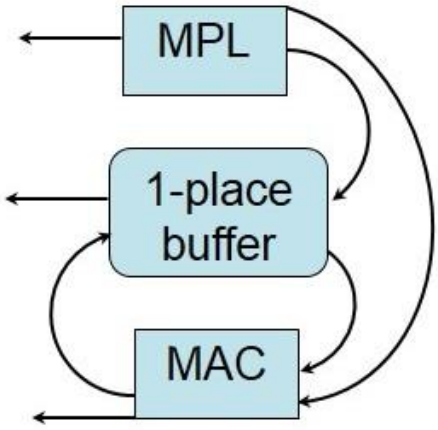

Figure 7, packet movements to MAC

When the MAC concludes the transmission, the WBP contains the packet returned by the MAC and the following algorithm is executed:

- When the WBP is successfully transmitted and the 1-place buffer is empty, do nothing.
- When the WBP is successfully transmitted and the 1-place buffer is filled, do nothing.
- When the WBP transmission failed and the 1-place buffer is empty, the WBP is passed to the 1-place buffer.
- When the WBP transmission failed and the 1-place buffer is filled, **choice B** is solved by the RT_i algorithm.

After filling the 1-place buffer and the MAC is free, **Choice C** is executed to choose between the 1-place buffers of the different seeds to select the packet to be passed to the MAC. Choice C is based on one of the following example policies:

1. Associate priorities to the seeds and take the packet from the 1-place buffer of the seed with the highest priority.
2. The packet with the largest hop count is chosen.
3. Use any of the other scheduling strategies published in literature such as presented in [5].

The scheduling algorithm, RT_i, determines the actions to take at choice A and choice B. The WBP and the 1BP have a deadline, W.dl and 1.dl respectively, and a sequence number, W.sq and 1.sq respectively. The comparison of the packets involves:

- When the seeds of the packets differ, a **choice C** is executed to choose between seeds.
- When the seeds of the packets are equal, the sequence numbers 1.sq and W.sq are compared in **choice A** and **choice B**.

Several scheduling algorithms, RT_i, are proposed when the seeds are equal. For every seed a 1-place buffer is reserved. The following scheduling algorithms are analysed with the simulator:

algorithm	condition	Choice A	Choice B
RT_0	no 1-place buffer	n.a.	n.a.
RT_1	n.a.	WBP -> 1-place buffer	Reject WBP
RT_2	n.a.	WBP -> 1-place buffer	WBP -> 1-place buffer
RT_3	n.a.	Reject WBP	Reject WBP
RT_4	n.a.	Reject WBP	WBP -> 1-place buffer
RT_5	WBP < 1BP	Reject WBP	Reject WBP
	WBP >= 1BP	WBP -> 1-place buffer	WBP -> 1-place buffer
RT_6	WBP < 1BP	Reject WBP	WBP -> 1-place, 1BP -> MAC
	WBP >= 1BP	WBP to 1-place buffer	WBP -> MAC

Table 2, choice of packet when 1-place buffer is full and new packet arrives

The RT_0 algorithm has no 1-place buffer, and the choices do not present themselves: the packet from MPL is submitted to the MAC directly and the return from the MAC is ignored. RT_6 is an optimization of RT_5, which is possible when packets of only one seed are forwarded. RT_5 is the so-called Highest Order First (HOF) algorithm.

7.2. Optimality of HOF

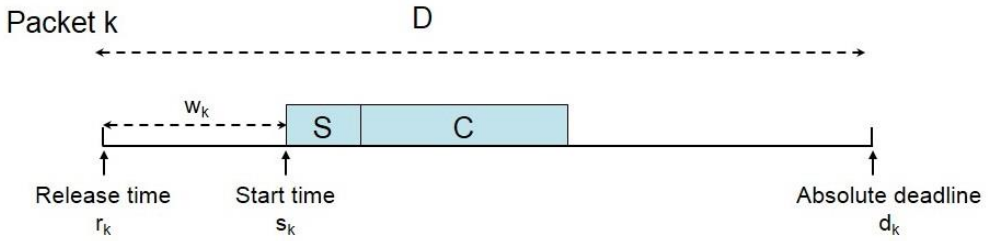

Figure 8, Packet timing model

HOF is shown to be optimal under overload. The packet model is shown in Figure 8. The packet, k, is started w_k time units after its release time, r_k. During interval S the MAC senses the medium. When the medium is occupied, the MAC signals no transmission. When the medium is empty, the packet is transmitted during an interval of length c. The packet meets ist deadline when $r_k+w_k+s+c < d_k$. Assume the sensing period << service time ($s < c$).

The HOF scheduling differs from LCFS of [6] in the following aspects:

- The highest order number is used instead of last arrived one (this is necessary because MPL repeats duplicates of packets).

- After sensing the medium to be busy, the MAC signals medium busy to MPL.
- HOF expects a fixed transmission time while LCFS handles a set of transmission times.

Packet p_k is the packet with order number k. The service time = s+c.

Theorem 1. When packets are periodically released with an interval between release times smaller than the service time, a relative deadline larger than the service time, and service starts immediately after transceiver liberation, the optimal scheduling strategy is Highest Order First (HOF) with a one-place buffer.

Proof Consider choice A, assume a HOF schedule. A given started packet, p_k, can either meet its deadline or not. Suppose packet p_k does not meet its deadline. Starting an earlier packet at the same moment as p_k was started will also lead to a deadline miss because the deadline of the earlier packet is smaller than the one of packet p_k. Suppose packet p_k meets its deadline. Starting an earlier packet at the same moment as p_k was started may lead to a deadline miss because the deadline of the earlier packet is smaller than that of packet k. Because the release period is smaller than the service time, a new packet is always released after starting a given packet. Consequently, starting the last released packet leads to at least as many packet transmissions which meet their deadline as any other choice of already released packets. Given that only the last released packet needs to be stored in the network buffer, a one-place buffer is sufficient.

Consider Choice B. When the packet is successfully serviced, the 1-place buffer is filled with a new packet. The new packet has been scheduled according to choice A and its deadline will be met. Suppose the packet is not serviced and no packet is present in the 1-place buffer. In that case the packet is restarted for transmission, if $w_k+c+s <$ deadline, and packet will meet its deadline.

Suppose the packet p_i is not serviced and a packet p_k is present in the 1-place buffer. Suppose $i < k$. In that case the deadline of p_k is larger than the deadline of p_i. Consequently, when p_k fails its deadline, p_i would also have failed; and choosing p_k is optimal.

Suppose the packet p_i is not serviced and a packet p_k is present in the 1-place buffer. Suppose $k < i$. Similar to the case $i < k$, the selection p_i is optimal. □

In addition, it must be proven that duplicate packets are not transmitted at the expense of the originals.

Theorem 2. When packets are periodically released with an interval between release times smaller than the service time, a relative deadline larger than the service time, and service starts immediately after transceiver liberation, the duplicate of p_k is only transmitted when for all p_i with $i<=k$ the originals of p_i have been transmitted

Proof Consider choice A. Suppose that the original p_k is overwritten in 1-place buffer. Due to the scheduling only packets p_{k+j} with $j>0$ can overwrite p_k. The earliest overwriting comes from p_{k+1}. In that case $r_{k+1} < r_k + s + c$. Packets are released with a fixed interval, from which it follows that for all j: $r_{j+1} < r_j + s + c$. Duplicates are released after the release of the original and can be stored in the 1-place buffer when

the 1-place buffer is free. Given that for all j: $r_{j+1} < r_j + s + c$, the 1-place buffer is never empty when the duplicate is released. Consequently, originals can only be overwritten by other originals.

Assume a replicate is executed by the MAC while an original p_k is overwritten by original p_m with m > k. The service time of the MAC is maximally s+c. Consequently $r_m < r_k + s + c$. Under choice A, replicate packets can be scheduled when $r_{k+1} > r_k + s + c$. This is a contradiction. □

Consequently, duplicates are not scheduled at the expense of originals.

7.3. Very small I_{min} values

The introduction of a 1-place buffer (and removal of queue in MAC) has consequences for the performance with small I_{min} values.

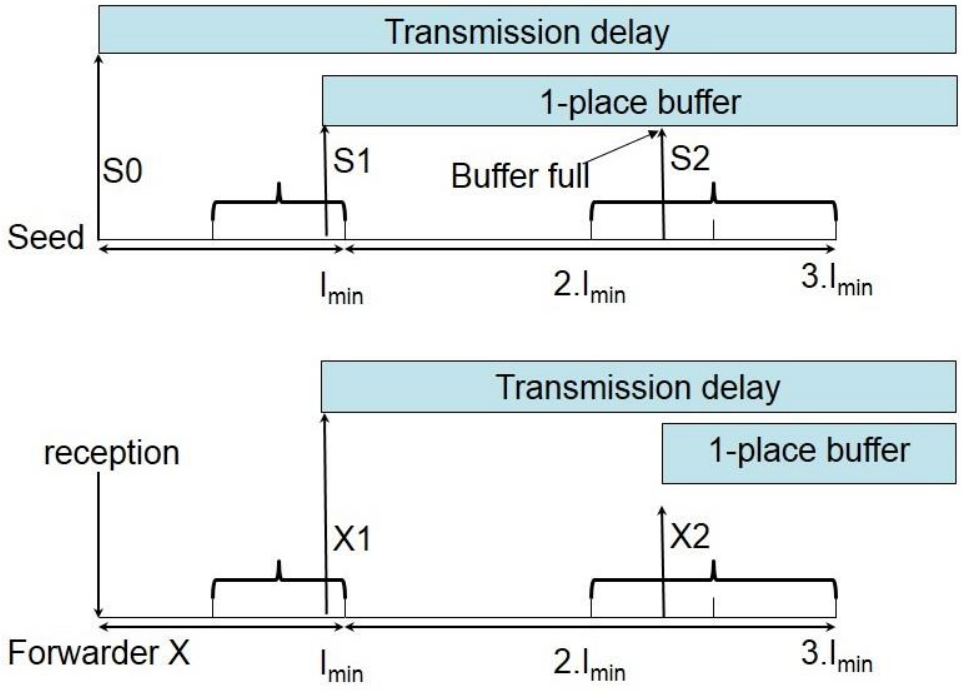

Figure 9, Loss of repeat 2 packet with small I_{min}

Imagine the sending of a message by the seed. Assume that I_{min} < delay/3 = 1.13 ms. First the packet, S0, is sent, followed by S1 and S2 packets before $3*I_{min}$ < delay. When packet S0 is in transmission, the S1 packet is stored in the 1-place buffer, but the S2 packet is refused, because the 1-place buffer is occupied and the S0 packet is in transmission. This situation is visualized at the top of Figure 9. When I_{min} > delay/2, the S2 packet can be stored in the 1-place buffer, because the S1 packet is in transmission and the 1-place buffer had been liberated.

Imagine the reception of a packet by a forwarder X and the sequential sending of the X1 and X2 packets. Counting from the moment of reception, the X1 packet is submitted to the MAC before I_{min}, and the X2 packet is submitted after $2*I_{min}$ time units. When packet X1 is still being transmitted at time = $3*I_{min}$, the X2 packet was stored in the 1-place buffer. The lower part of Figure 9 visualizes the repeater scenario.

7.4. Large I_{min} values

Figure 10 visualizes the behaviour for large I_{min} values. The behaviour of the seed is different from the forwarder.

Consider the seed with a deadline equal to the minimum of the generation interval or 200. When I_{min} < generation interval <2* I_{min}, S1 packets are sent before their deadline but all S2 packets miss their deadline. When I_{min} > 2*generation interval all S1 packets and S2 packets miss their deadline. When $3*I_{min}$ < generation interval, all S1 and S2 packets can meet their deadline.

Consider the forwarder with the same deadline. All X2 packets generated by the forwarder miss their deadline when I_{min}> dl/2. A subset of the X2 packets generated by the forwarder miss their deadline when dl/3 < I_{min} < dl/2. When I_{min} > 2*dl, all X1 and X2 packets miss their deadline.

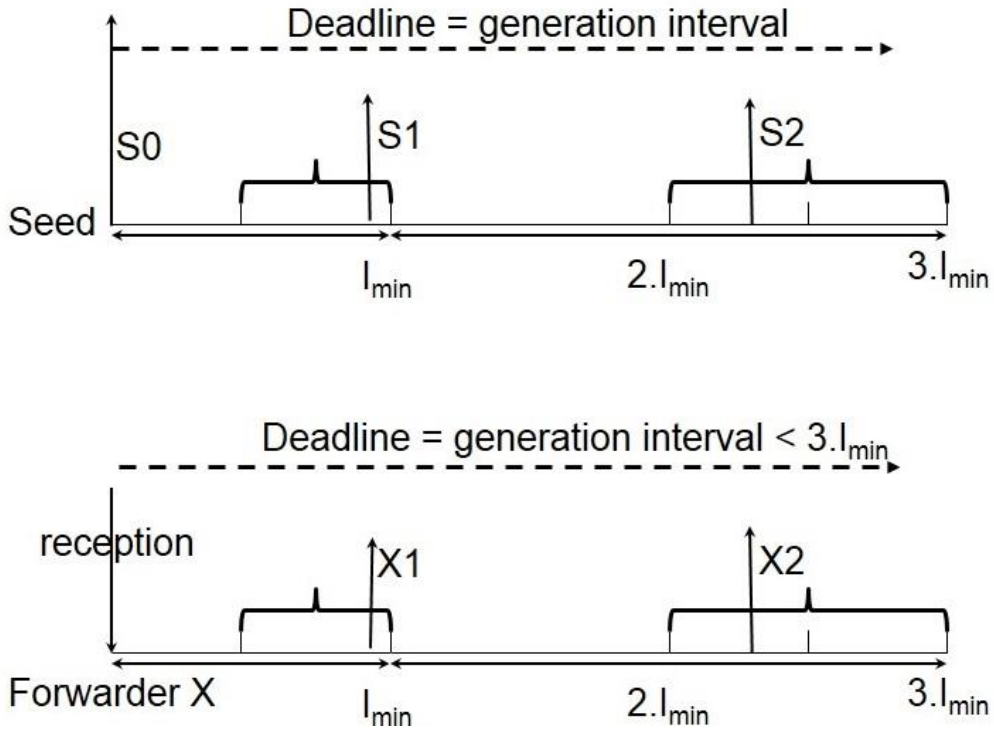

Figure 10, Packet behaviour for large I_{min},

8. Simulation conditions

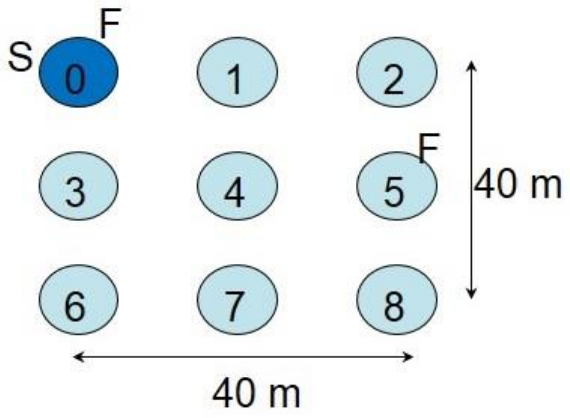

Figure 11, 1-hop mesh configuration

Simulation is done for two types of configuration: (1) one hop network (e.g. one office) and (2) multi-hop network (e.g. open space offices).

Figure 11 shows the mesh configuration used for the one-hop network. The distance between the nodes is maintained at 20 meters. The network is chosen such that all nodes can reach all other nodes in one hop. This situation is representative for many real life installations (one office with a presence sensor and lights). Node 0 is chosen as seed (denoted with S). The probability that a packet is lost during transmission can be set (0 – 20%). In addition, it has been observed that multi-path fading leads to the disappearance of the channel between any 2 nodes during a relatively long time (seconds to minutes). In the simulation this is modelled by having node 4 refuse packets from node 0 during 8% of the time in intervals of tens of seconds. Nodes 5 and 0 are chosen as forwarder (denoted with F). Node 0 is a forwarder and repeats packets to make sure that every packet is received by at least one neighbour node.

Figure 12 shows the mesh configuration for the multi-hop network. This is composed of a series of 1-hop networks. The distance between the nodes is maintained at 20 meters. Local seeds (nodes 3 + n*9, n=0 to 4) send messages destined to the local 1-hop neighbours. The multicast group of sender 3 is composed of nodes 0 to 9, the multicast group of sender 12 is composed of nodes 9 to 17 and so on. One global seed (node 1) generates messages for all nodes (0-44) in the network. The forwarders used for the 1-hop network also serve as forwarder for the n-hop network.

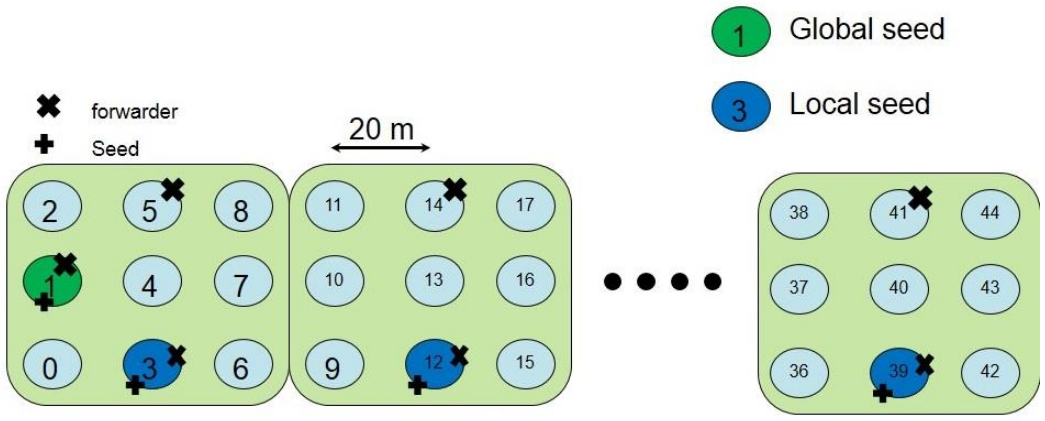

Figure 12, multi-hop mesh

The channels between the forwarders for the n-hop distribution from the global seed are shown in Figure 13.

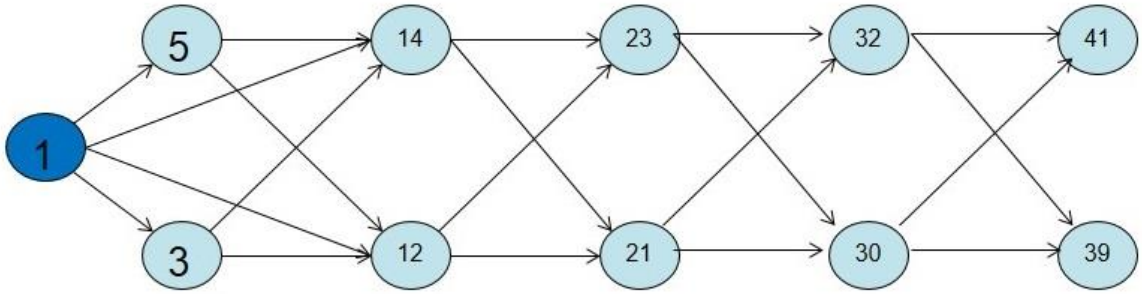

Figure 13, channels between forwarders for packets sent by global seed.

Seeds generate messages with a fixed *generation interval* per simulation, ranging from 10 ms to 400 ms. A jitter of 10% is imposed on the interval generation to remove artefacts due to the strict simulation timing

9. Simulation results

9.1. Real-time scheduling

The simulation quantifies the improvement that comes from scheduling packets in the network layer without packet buffering and multiple back-offs in the MAC (see section 7.1). Scheduling algorithms are compared for k=4, I_{min}=40 ms, and the generation interval goes from 10 ms to 100 ms. in 10 ms steps.

9.1.1. 1-hop

We investigate the 1-hop network with 9 nodes of Figure 11. Simulations have been done to determine the end-to-end delay and the percentage of loss for node 4. Node 0 is always the seed. The channel between node 4 and the seed is unavailable during 8% of the time. With 3 packets per seed, and one forwarder with 2 packets, the load per generation interval is 5 packets times 3.4 ms results in 17 ms. This constitutes an overload situation with generation interval < 20 ms. The simulations give an idea about the maximum end-to-end delay, and the losses which are suffered at node 4. The algorithms of section 7.1 are used. Algorithm RT_0 uses standard MAC buffer of 3 packets and no 1-place buffer. Algorithms RT_1 to RT_6 use a 1-place buffer and no buffer in the MAC.

Losses in number of packets, with 20,000 packets sent, are displayed for ten generation intervals in Table 3. The results of algorithms rt_1 to rt_4 are quite close and perform half way between rt_0 and rt_6. Results of rt_1 to rt_4 are not shown in the following sections for that reason.

Removing the MAC buffer significantly reduces losses especially for rt_5 and rt_6, as intended.

Generation interval	rt_0	rt_1	rt_2	rt_3	rt_4	rt_5	rt_6
10	5213	2162	2218	2236	2368	237	116
20	153	52	50	47	59	5	1
30	1	12	6	12	9	7	1
40	1	2	0	7	1	0	0
50	0	0	0	0	0	0	0
60	0	1	0	0	0	0	0
70	0	1	1	1	1	0	1
80	0	0	2	0	2	0	0
90	1	1	0	0	1	0	0
100	0	0	0	0	0	0	0

Table 3, packets lost for generation intervals and scheduling algorithms

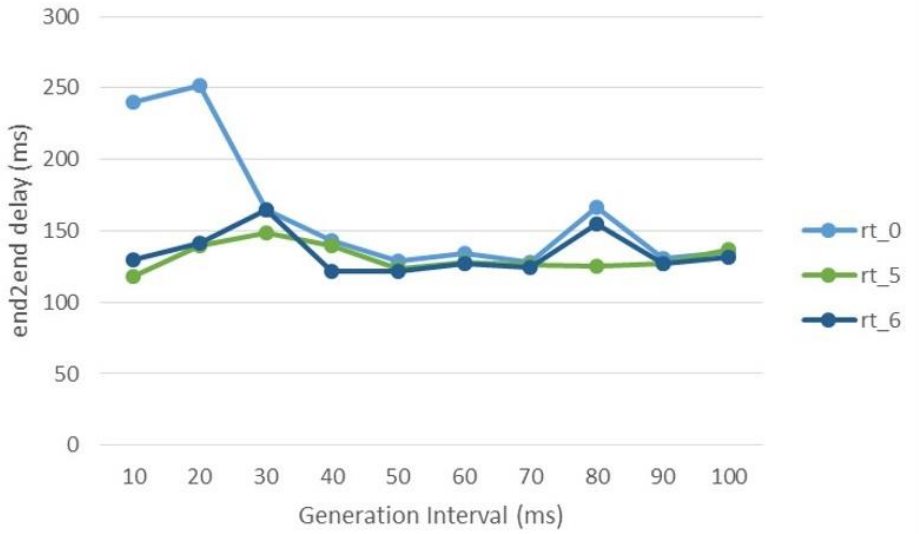

Figure 14, maximum end-to-end delay (ms), at node 4 for 3 scheduling algorithms as function of generation interval (ms)

Figure 14 shows the end-to-end delay at node 4 expressed in ms as function of the generation interval in ms for 3 different scheduling algorithms. The figure shows that the maximum end-to-end delay is largest without real-time scheduling (rt_0). The presence of the MAC buffer increases the maximum delay with 150 ms for generation interval = 10, 20 ms. The influence of the MAC buffer decreases with increasing generation interval because packets can be transmitted without back-offs, and the buffers in the MAC are not used.

For the 3x3 configuration, the rt_5, rt_6 algorithms provide lowest losses and a lower maximum end-to-end delay. From a loss and the maximum end-to-end delay perspective, real-time scheduling algorithms have a clear advantage over no scheduling.

9.1.2. N-hop

Scheduling of the communication in the n-hop network with 45 nodes (see Figure 12) is analysed in this section. Simulations have been done to determine the end-to-end delay and the percentage of loss for node 44. Node 44 is representative for the worst case behaviour in the network. Node 1 is the seed. No communication losses are considered. The network supports 10 forwarders, where the pairs (12, 14), (21, 23) and (30, 32) are the forwarders to node 44 from node 1 (see Figure 13). The pair (3, 5) adds to the network load at node pair (12, 14). These forwarders are necessary to tolerate the network losses investigated later (see sections 9.4 and 9.2). The highest packet load occurs around forwarder pair (12, 14). Per message, 3 packets arrive from the seed, and 2 packets from the surrounding 5 forwarders. The load at node 12 or node 14 per generation interval is 13 packets times 3.4 ms results in 44.2 ms. This constitutes an overload situation with generation interval < 50 ms. The simulations provide the maximum end-to-end delay and the losses which are suffered at node 44. The algorithms of section 7.1 are used.

The performance with a MAC buffer is worse than without MAC buffer and confirms the results for the 1-hop case of section 9.1.1.

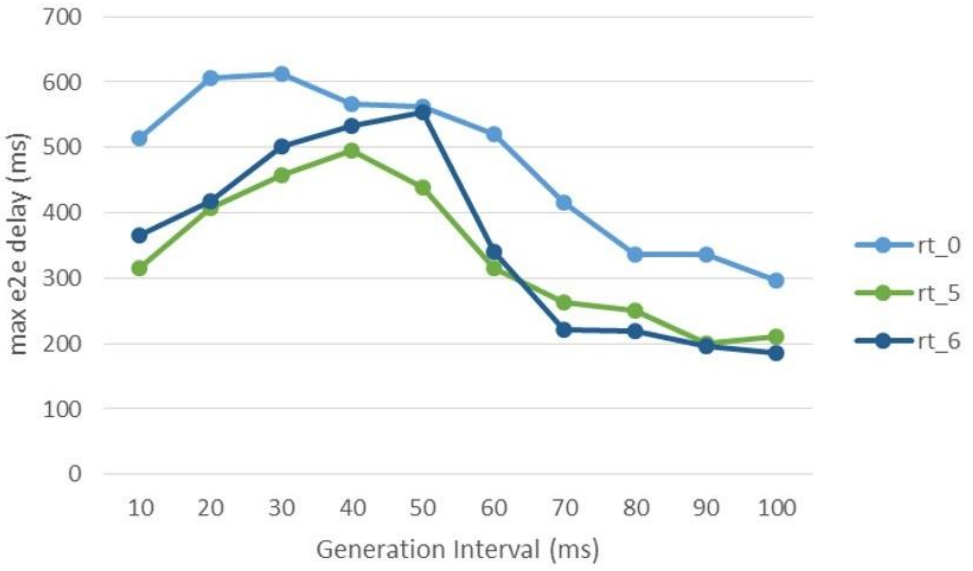

Figure 15, Maximum end-to-end delays (ms) at node 44 as function of generation interval (ms) for 3 scheduling algorithms

Figure 15 shows the maximum end-to-end delays in ms as function of the generation interval in ms for 3 scheduling algorithms at node 44 with I_{min} = 40. The figure shows that the maximum end-to-end delay is smallest with rt_5 and r_6. Beyond a generation interval of 60 ms the maximum end-to-end delay of rt_5 and rt_6 are around 250 ms.

Figure 16 shows the losses, expressed in packets lost, as function of the generation interval in ms.

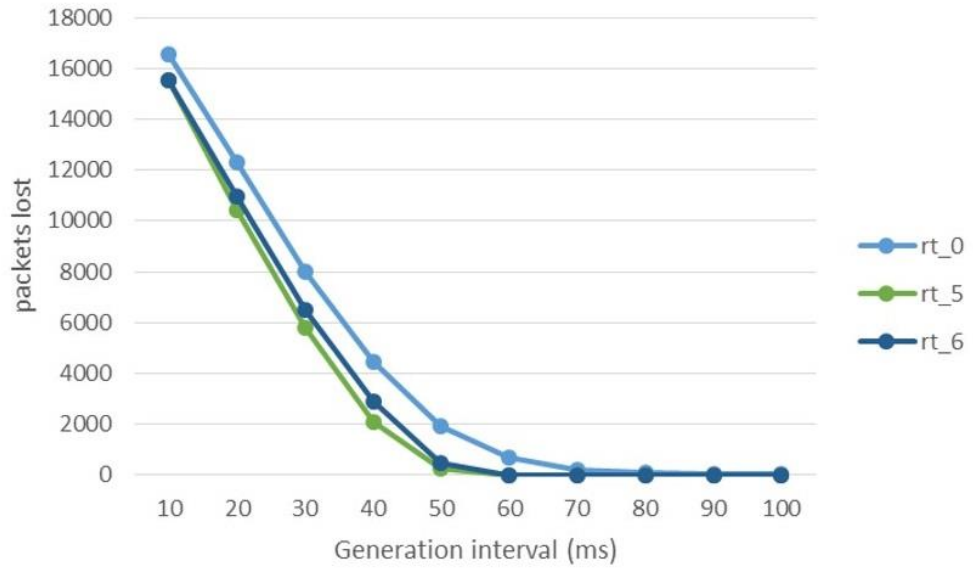

Figure 16, losses in number of packets for different scheduling algorithms as function of generation interval (ms) at node 44

For the generation interval larger than 60 ms both rt_5 and rt_6 do not lose any packets.

9.1.3. Choice of scheduling algorithm

The rt_5 algorithm is chosen, given its superior performance for both the 3x3 and the 3x15 meshes. Algorithm rt_6 is performs better than algorithm rt_5 but relies on an optimization that is possible when packets of only one seed are forwarded.

9.2. Forwarder configuration

This section shows that forwarders are necessary to tolerate packet losses during point to point transmission. The scheduling algorithm rt_5 is used, and other parameter values are: k=4, I_{min} =40 ms, and generation interval = 200 ms. The latter value is chosen large to remove any loss coming from overload. Transmission losses are introduced as function of packet communication per hop losses in percent: 0%, 5%, 10%, 15%, and 20%.

9.2.1. 1-hop mesh

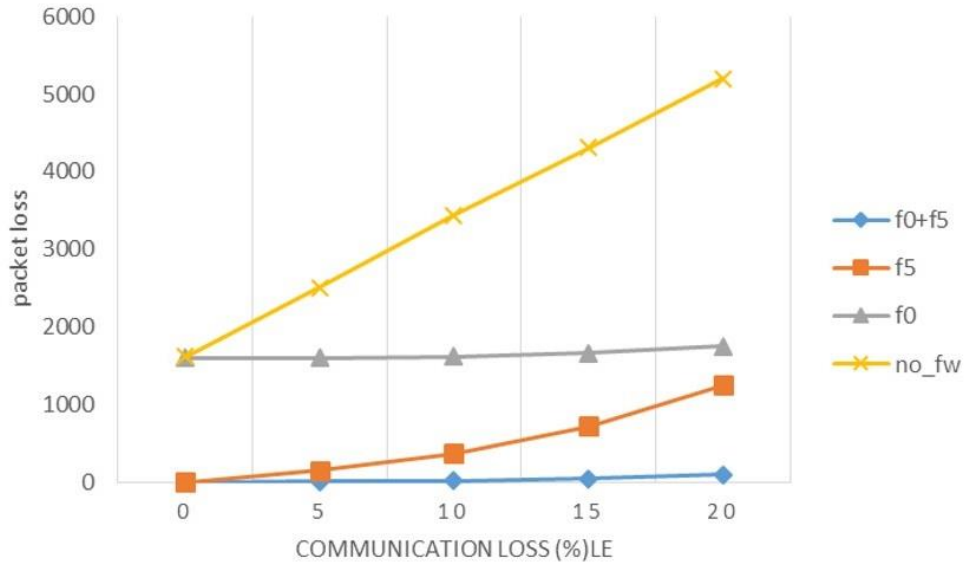

Figure 17, message loss as function of packet transmission loss in %, for 4 forwarder configuration

Consider the 3x3 mesh of Figure 11. The percentage of loss for node 4 as function of the forwarder configuration is analysed. Node 4 is the key node because the channel between node 0 (seed) and node 4 is non-existent during 8% of the time. Node 0 is always the seed.

Four forwarder configurations (see Figure 11) are considered:

- no forwarder is present in the mesh (no_fw)
- node 5 is forwarder (f5)
- node 0 is forwarder (f0)
- nodes 0 and 5 are both forwarders (f0+f5)

Figure 17 shows the loss (expressed as number of packets) at node 4 as function of the 1-hop communication loss in percent (horizontal axis) for the 4 forwarder configurations. The total number of packets sent is 20,000. For f0+f5 with a 20% transmission loss, 99 of the 20,000 (5 permil) messages do not arrive. The losses for transmission loss 0% come from the blocking of the channel between node 0 and node 4 for 8% of the time. The losses with f5 and f0+f5 configurations are zero for transmission loss 0%.

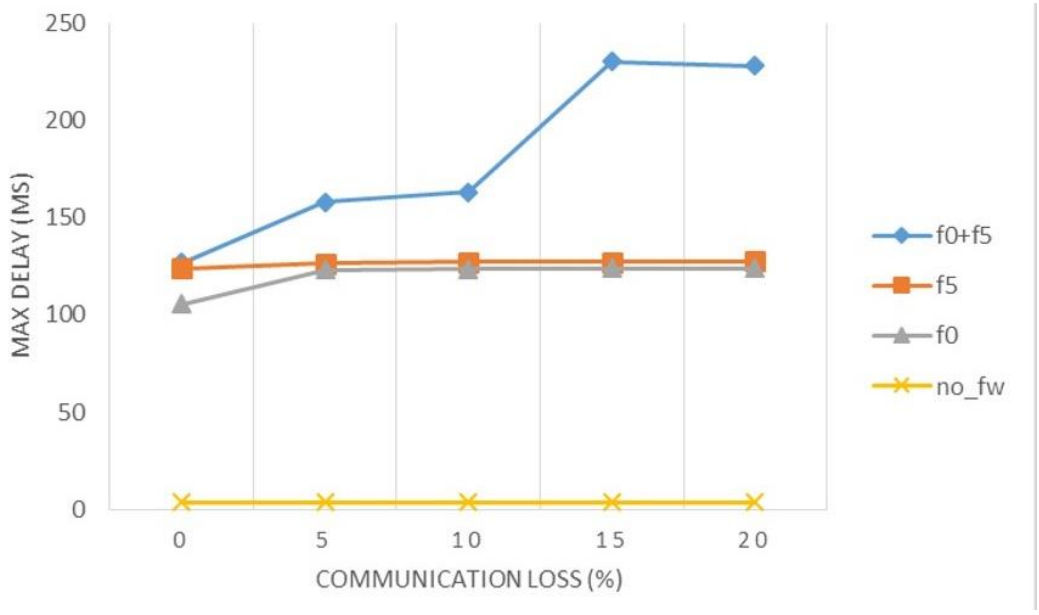

Figure 18, maximum end-to-end delay (ms) as function of transmission loss percentage

Figure 18 shows the maximum end-to-end delays (ms) at node 4 as function of transmission loss (in percent) corresponding with the four forwarder configurations. The lowest maximum end-to-end delay is measured for no forwarder present, corresponding with direct 1-hop communication. The largest maximum end-to-end delays occur for forwarder 0 and 5 configuration. The simulation shows that these larger values come from the repetition by node 5 and 0 of the packets that would otherwise be lost. The max delay of 120 ms for f5 and f0 configuration corresponds with $3*I_{min} = 3*40 = 120$ ms. The value of 240 ms corresponds with the addition of twice $3*I_{min} = 2*3*40 = 240$ ms.

Conclusion is that a forwarder at source and an additional forwarder are necessary for the fault assumptions described here. In the presented 1-hop scenarios the maximum delays are contained within 250 ms. The following sections will show how the end-to-end delay decreases with decreasing I_{min}.

9.2.2. N-hop mesh

The forwarder configuration of Figure 12 is used. No long duration channel losses between seed and a given node were introduced in the simulation. Table 4 shows the losses, average and the maximum end-to-end delay for the messages arriving at node 44.

Transmission loss	Avg end-to-end delay	Max end-to-end delay	Messages lost
0 %	106 ms	179 ms	0
5%	108 ms	185 ms	0
10%	110 ms	214 ms	0
15%	113 ms	259 ms	0
20%	117 ms	315 ms	3

Table 4 , node 44, avg, max end-to-end delay and number of lost messages

Losses are less than .1 permil, and the end-to-end delay stays around 200 ms. The table results show that enough forwarders are present in the 15x3 configuration. The maximum delay increases because with increasing loss the probability increases that the first packet is lost or that more hops are involved. The following sections will show how the end-to-end delay decreases with decreasing I_{min}.

9.2.3. Forwarder choice

Choosing nodes 0 and node 5 as forwarders in 3x3 mesh provides losses of less than 4 permil and maximum end-to-end delays below 250 ms with a transmission loss probability of 20%.

Choosing the pairs of forwarders in 3x15 mesh provides losses of less than 0.1 permil and maximum end-to-end delays below 315 ms with a transmission loss probability of 20%

9.3. Choosing I_{min} and k

Section 5.3 explains how the end-to-end delay for a multi-hop network is related to the I_{min} value. An I_{min} value lower than 40 should decrease the end-to-end delay to a value below 200 ms. As explained in section 6.1 the presence of forwarders leads to an increase of c. The value that c will reach depends on the value of I_{min} and the number of forwarders.

The loss caused by an increase of c as function of I_{min}, and the dependence of the end-to-end delay simulations are shown with I_{min} =10, 20, 30, and 40 ms. A value of I_{min} > 20 allows an increment of c by 5

consecutive packets (5*3.4 = 17 < 20). A generation interval of 200 ms is chosen to avoid overload losses. No packet transmission losses are included.

9.3.1. 1-hop mesh

In the 3x3 mesh node 0 and 5 are chosen as forwarders as recommended in section 9.2.3. In the 3x3 mesh, packets from seed 0 can increase c in forwarder 5, leading to a maximum c value of 1 (see section 6.1). In the simulation, the channel between seed and node 4 is closed for 8% of the time. The probability, P, that the packet sent by node 0 increases c in node 5 for X1 packet, is given by: P = .5. The probability that this happens for both X1 and X2 packets is 0.25. Consequently there is a probability of 0.25 that node 5 does not repeat the packet at all and node 4 will receive nothing when the channel between node 0 and node 4 is closed. Table 5 shows losses at node 4 for k=1. For k=2 and 3 the loss is zero (not shown in the table).

	I_{min} = 10	I_{min} = 20	I_{min} = 30	I_{min} = 40
Lost packets	391	382	483	416

Table 5, packets lost at node 4 for different I_{min} values, k=1

The loss of around 2% is indeed a quarter of the loss of 8% suffered by node 4 when node 5 is not a forwarder and independent of the I_{min} value.

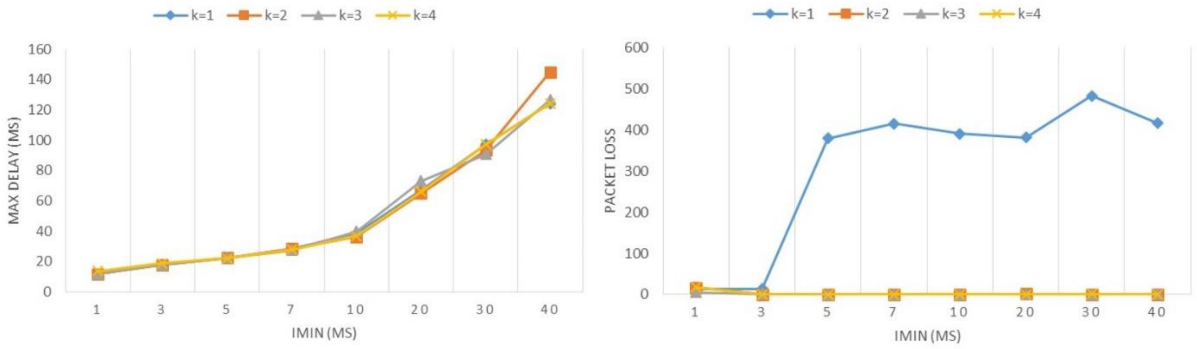

Figure 19, maximum end-to-end delay and loss at node 4 as function of I_{min} and k= 1, 2, 3, 4

Figure 19 shows the maximum end-to-end delays at node 4 as function of I_{min} (ms) for 4 k values. These maxima are associated with packet S0 from the seed to forwarder 5 followed by the packet X2 from forwarder 5 to node 4 for I_{min} > 10. (Beware axis changes at I_{min} = 10 ms). The maximum end-to-end delay from the seed to node 5 by packet S1 is given by the transmission time + trickle timer interval, which amounts to I_{min} + 3.4 ms. The average contribution of packet X2 from forwarder 5 to node 4 is given by 2*I_{min} + 3.4 ms. The total is 6.8 ms + 3*I_{min} (Values are: 37, 67, 97, 127, which corresponds with the simulated values). The losses for k=1 reduce to zero when I_{min} < 5. As explained in section 6.1 this comes because packet X1 is already offered to the MAC before S1 has finished transmission.

9.3.2. N-hop mesh

Figure 20 shows losses (number of packets) as function of node number for k=1 to k=4, and I_{min}= 40. In the figure, packets are progressively lost after every repetition. Packets, arriving in node 44, have been repeated consecutively by forwarder pairs (12, 14), (23, 21), and (32, 30) (see Figure 13). Failure to propagate packets in pair (12, 14) shows up as losses at node 16 and higher. Messages from seed 1, forwarders 3, 5, and 12 can increase c in forwarder 14, leading to a maximum c value of 4. In the same manner seed 1, forwarders 3, 5, and 14 can increase c in forwarder 12. For that reason losses occur for k<4. Apparently, c can reach the value 3 in forwarder pair (12, 14). This increment of c is caused by the packets arriving from the seed and the forwarder pair (3, 5). Removing the forwarder pair (3, 5) led to no losses for k > 1.

The figure also shows that with k=2, no extra losses occur at pairs (23, 21) and (32, 30) indicating that only one forwarder increments c at these pairs. For k=4 no losses occurred. For k=3, two to three packets were lost for I_{min} = 40.

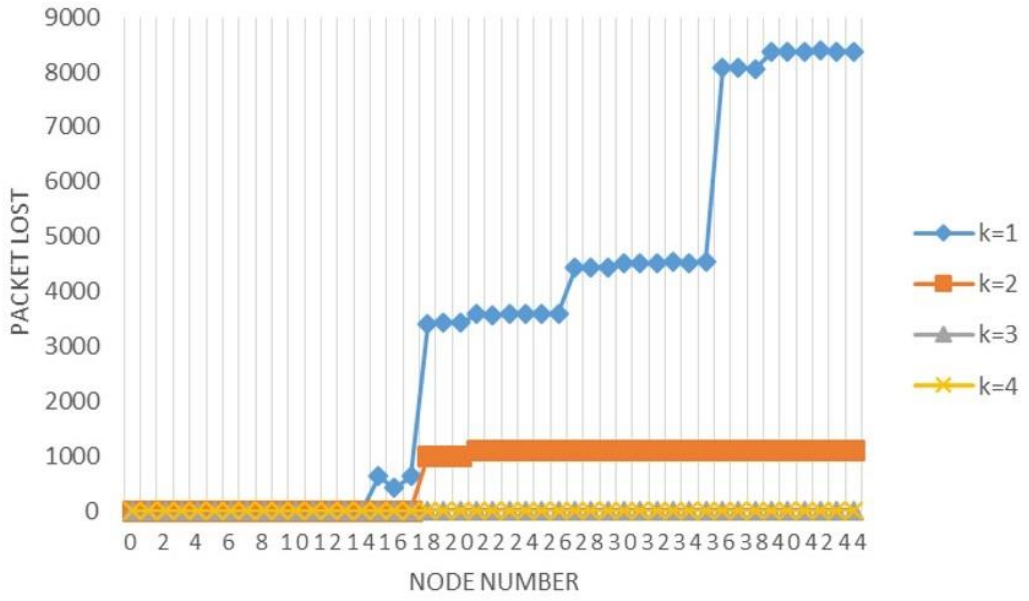

Figure 20, number of lost packets as function of node number for four k values, and I_{min}=40

The losses at node 44 as function of I_{min} for four k-values are shown in Figure 21. As predicted in section 6.1, higher I_{min} values increase the probability that c is incremented when several forwarders repeat a packet. For k=3, 4 and I_{min} > 10, losses are zero or 2-3 packets on 20,000. For k=1, the loss stabilizes at I_{min} =20. Loss decreases for I_{min} > 20, and k=1, because the probability that c is increased beyond 1 decreases for packet X1. The probability that c is increased beyond 2 increases for packet X2 for I_{min}>20 leads to higher loss for k=2.

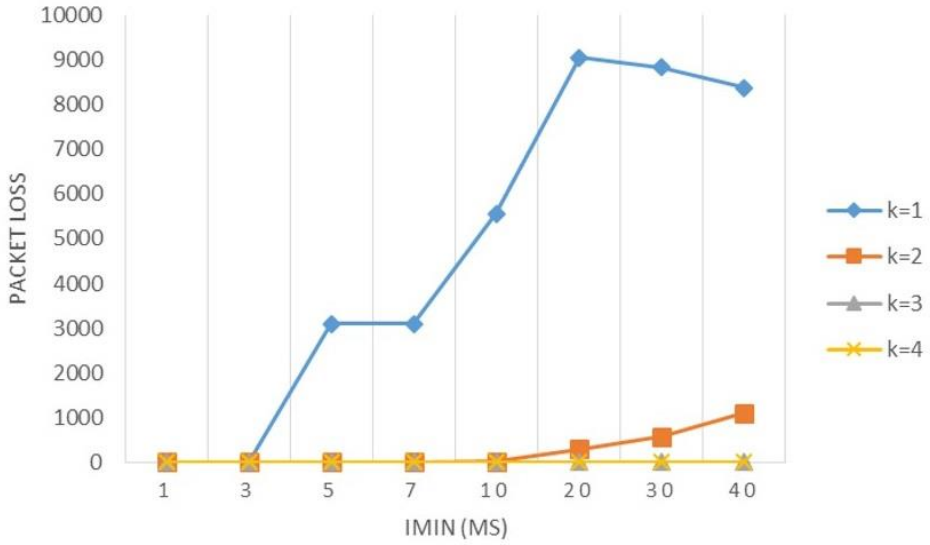

Figure 21, number of lost packets at node 44 as function of I_{min} for four k-values

The maximum and minimum end-to-end delays depend on the hop count and I_{min} according to section 5.3. For the remainder of the simulations in this section we use k=4 to avoid losses from the c > k condition.

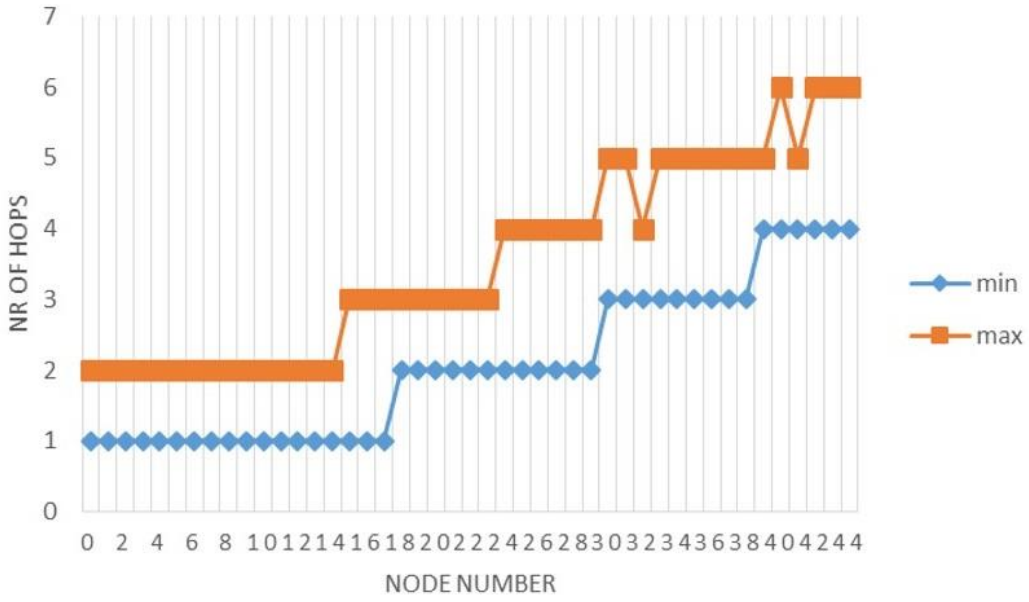

Figure 22, minimum and maximum hop count as function of node number for I_{min} = 10, k=4

Figure 22 shows the minimum and maximum hop counts of the packets as function of the node number.

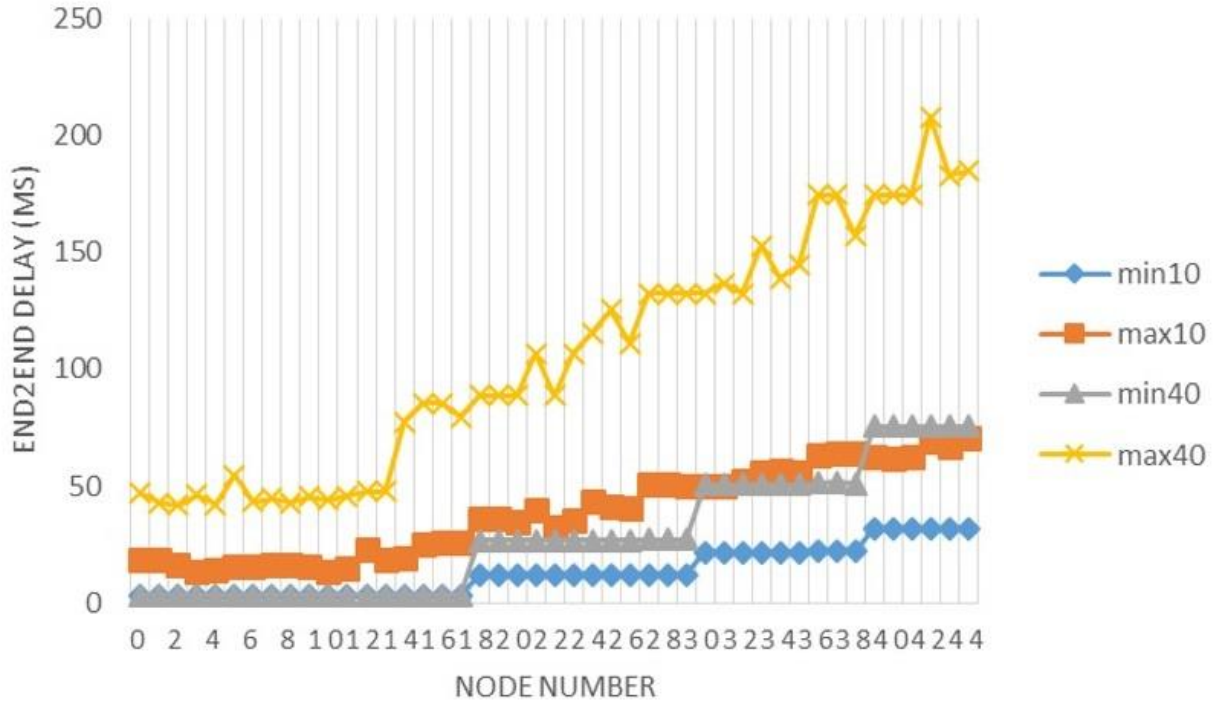

Figure 23, maximum and minimum end-to-end delay (ms) as function of node number for I_{min} =10, 40 , and k=4

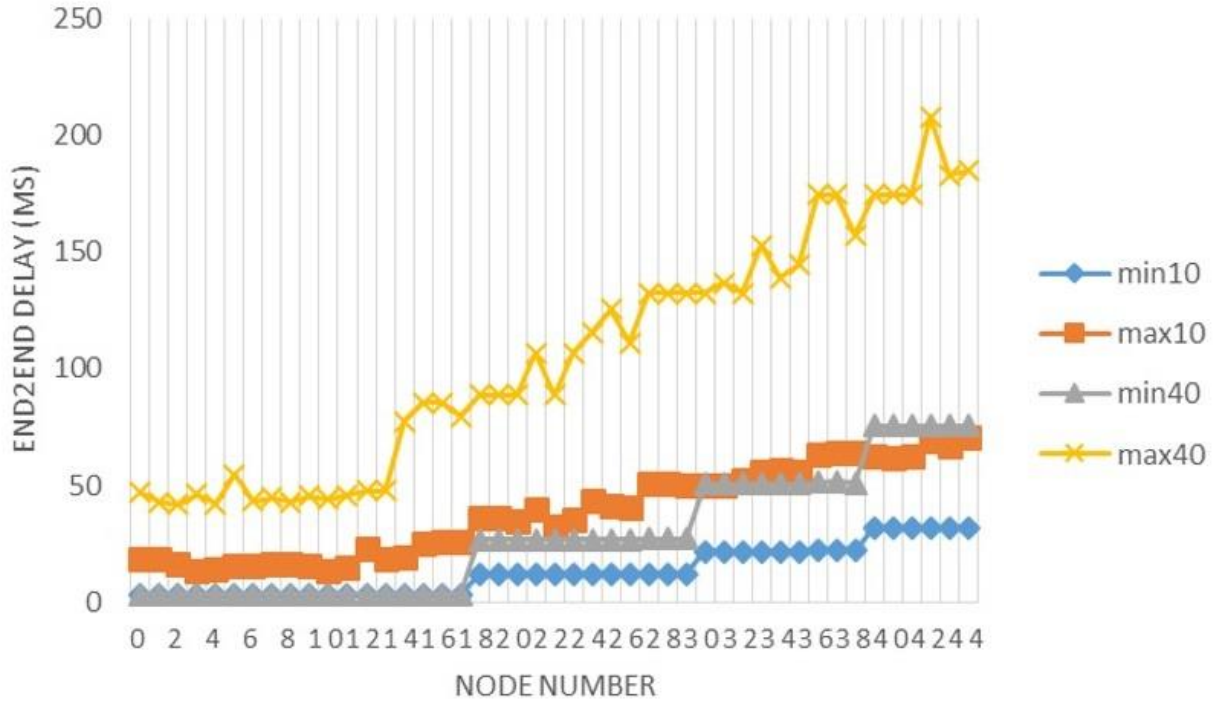

Figure 23 shows the corresponding maximum and minimum end-to-end delays (ms) as function of node number for I_{min} =10, 40, with k=4. The minimum end-to-end delays are clearly increased with the hop count. The maximum hop count is less clearly visible in the maximum end-to-end delay. Next to the hop

count, the maximum end-to-end delay is related to the loss of packet X1 or long wait times in 1-place buffer.

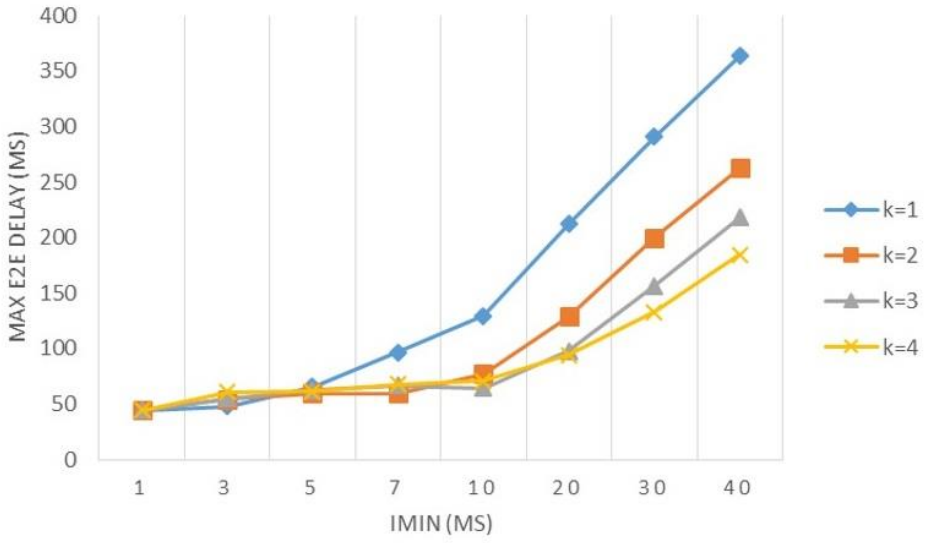

Figure 24, Max delay at node 44 as function of I_{min} and for 4 k values

Figure 24 shows the evolution of the maximum delay at node 44 as function of I_{min} for four k values.

The maximum end-to end delay (Emax) and the minimum end-to-end delay (Emin) can be expressed as function of transmission time (d), minimum hop (minh), maximum hop (maxh), and I_{min} (I), assuming packet X1 succeeds on all hops and packet X2 is not needed.

```
Emax = d + (maxh-1)*(max(d, I)+ d)
Emin = d + (minh-1)*(max(d, I/2)+ d)
```

The term d comes from the first hop, while the term including I_{min} comes from the following hops (see section 5.3). Table 6 compares the calculated values with the simulated values for node 44. Maxh is taken 6 and minh is taken 4 from Figure 22. The max (d, I/2) and max (d, I) are needed because of the three conditions on transmission delay and I_{min} of section 6.1. In this case: $I_{min}/2 > d$.

	I_{min} = 10		I_{min} = 20		I_{min} = 30		I_{min} = 40	
	Emin	Emax	Emin	Emax	Emin	Emax	Emin	Emax
simulation	32	70.6	45	94	60	133	75	184
calculation	28.6	70.4	43.6	120.4	55.2	170.4	73.6	220.4

Table 6, calculated and simulated end-to-end delays for node 44, k=4

Table 6 shows the measured- and calculated minimum and maximum end-to-end delays, assuming X1 is the first packet to arrive. As expected the minimum simulated end-to-end delays are larger than the minimum calculated ones. Maximum calculated values are larger than their simulated counter-parts. For I_{min} = 10, packet X1 does not always arrive, and the maximum end-to-end delays are increased with $2*I_{min}$, possibly for all hops. When packet X1 is lost, packet X2 determines the end-to end delay. The corresponding maximum Emax2 amounts to:

```
Emax2 = d + (maxh-1)*(max(2*d, 3*I) + d)
```

With packet X2, the maximum end-to-end delay increases with 2*(maxh-1)*I, (respectively, 100, 200, 300, and 400 ms). These values are simulated for k=1, when the probability of losing packet X1 has increased.

9.3.3. I_{min} and k choice

The sections 9.3.1 and 9.3.2 show that the end-to-end delay for the 3x15 mesh decreases with decreasing I_{min}, and k > 3 avoids losses coming from the c>k condition in the forwarders.

9.4. Multiple senders

In this section the behaviour of the 3x15 mesh with multiple senders is analysed. It is shown that care should be taken to activate a minimum number of forwarders. The example mesh is shown in Figure 12. The 15x3 mesh is subdivided in five 3x3 meshes. Each 3x3 mesh contains one seed which is also forwarder, and one additional forwarder. The seeds in the 3x3 mesh send their messages to different "local" multicast addresses. All nodes in the same 3x3 mesh are enabled to receive messages for the local multicast address used by the seed in that 3x3 mesh. Node 1 sends messages to a "global" multicast address. All nodes in the 15x3 mesh are enabled to receive messages for the global multicast address. All senders start at a random moment within the generation interval. The RT_5 algorithm is used throughout.

In this section two things are discussed:

- The importance of reducing forwarder load
- Consequences of transmission errors

This section shows for fixed k=4 the effect on end-to-end delay of I_{min} under different loss conditions (4%, 12% and 20%). In the 3x15 mesh node 1 "globally" generates packets for all nodes, and nodes 3, 12, 21, 30 and 39 "locally" generate packets for their 8 neighbours (see Figure 12). Generation interval is set to the same value for all nodes to avoid losses coming from overload conditions. Section 9.4.2 discusses the global behaviour under transmission losses, followed by section 9.4.3 which discusses the local behaviour under transmission losses. Section 9.4.1 discusses the need to minimize the number of forwarders.

9.4.1. Forwarder configuration

Simulation was done for two cases:

1. Forwarders repeated packets for all multicast addresses (all-forwarder)
2. Forwarders repeated only packets for their own domain (domain-forwarder)

I_{min} is chosen 10 ms and k=4. Node 22 reflects reasonably well the performance of the other nodes. For this node the packet loss in percent is shown in Figure 25. The x-axis represents the value of the generation interval in steps of 40 ms.

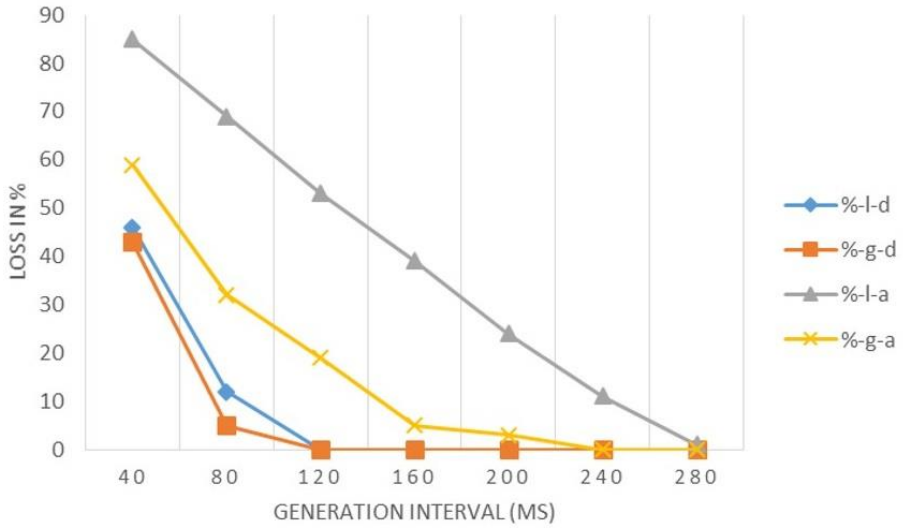

Figure 25, packet loss in percent at node 22 as function of generation interval for all forwarders and domain forwarders

The following graphs are shown in Figure 25 :

- %-l-a: percent lost for local multicast using all forwarders
- %-g-a: percent lost for global multicast using all forwarders
- %-l-d: percent lost for local multicast using domain forwarders
- %-g-d: percent lost for global multicast using domain forwarders

The local losses are larger than the global losses because packets with high hop-counts are favoured by RT_5. In node 22 all global packets execute 1 hop or more. Consequently, local packets are rejected in favour of global packets.

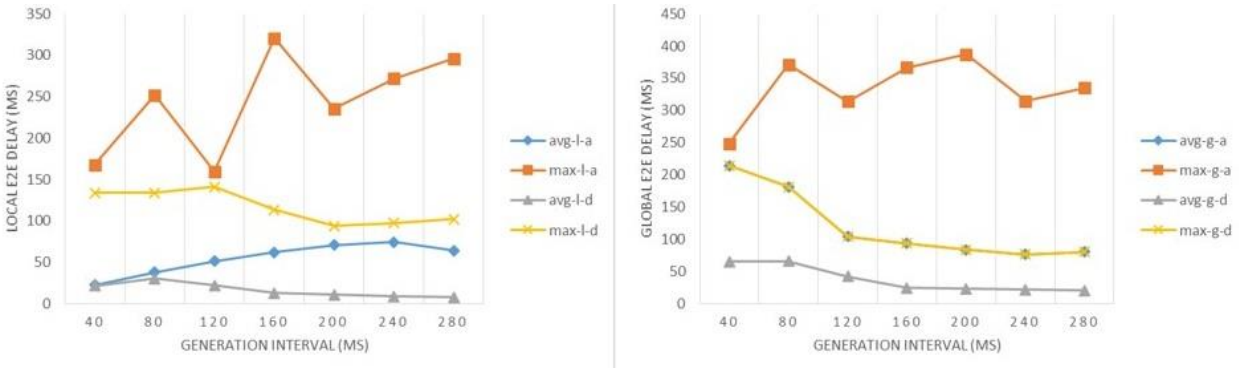

Figure 26, average and maximum end-to-end delay (ms) as function of generation interval (ms) at node 20; left local multicast, right global multicast

Figure 26 shows the end-to-end delays (ms) as function of the generation interval (ms). The figure shows that the average (avg-l-a) and the maximum (max-l-a) end-to-end delays for the local multicast are much larger when all forwarders repeat all messages than the average (avg-l-d) and the maximum (max-l-d) end-to-end delays when forwarders only repeat messages for their domain. With small generation interval the load is heavier, communication takes longer, and packets are removed because they exceed their deadline.

For the global multicast also the delays are longer when all forwarders participate.

Conclusions are:

- Reducing the number of forwarders reduces the network load. For local multicasts, only forwarders belonging to the local domain should repeat local multicast packets.

9.4.2. Global multicast

The global multicast performance with the presence of local multicast resembles the performance of the single global multicast without local multicast described in section 9.3.2. With a 20% point to point communication loss, only 10 packets on 20,000 are lost in nodes 39-44. The following parameter values are fixed: generation interval = 400 ms, and k=4.

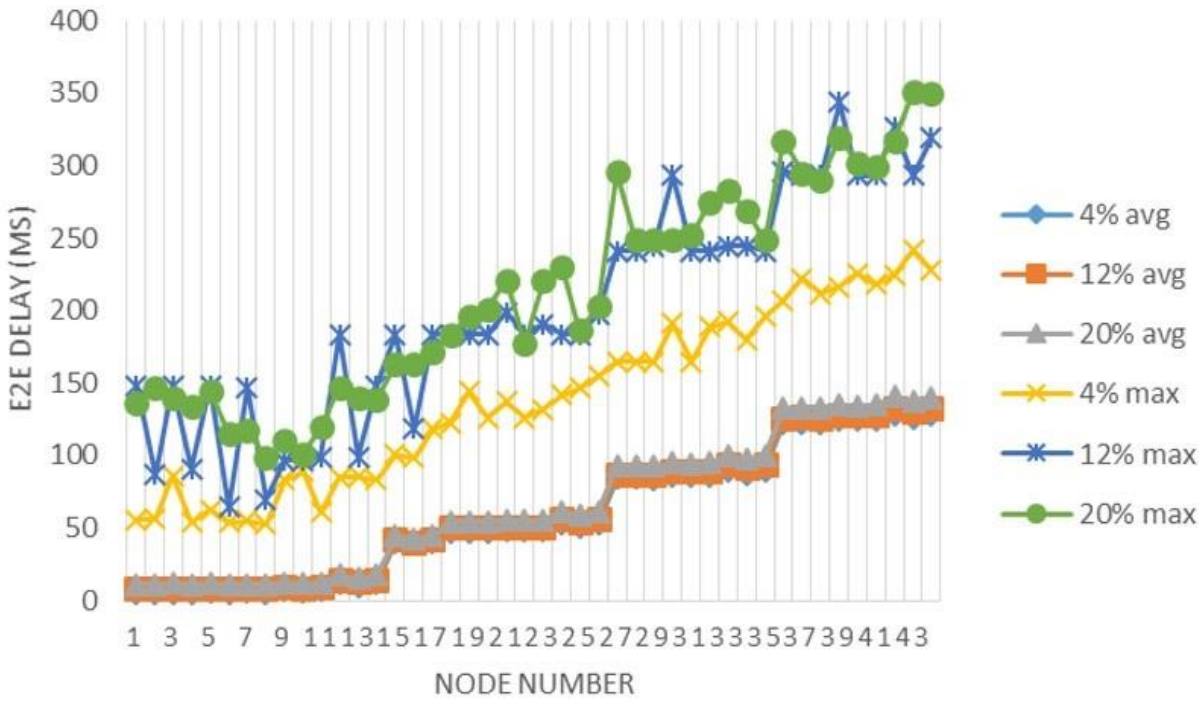

Figure 27, global end-to-end delay as function of node number for 3 communication error probabilities, I_{min} = 50

Figure 27 shows the end-to-end delay of the global multicast as function of node number for three communication error probabilities (4%, 12% and 20%). The parameter I_{min} = 30, which is shown to be a good value in the next section. With communication error of 4% the maximum is determined by packets X1. With higher error probabilities, packet X2 has a higher impact on the maximum end-to-end delay. The average end-to-end delay remains within 150ms, but the maximum end-to-end delay reaches almost 350 ms.

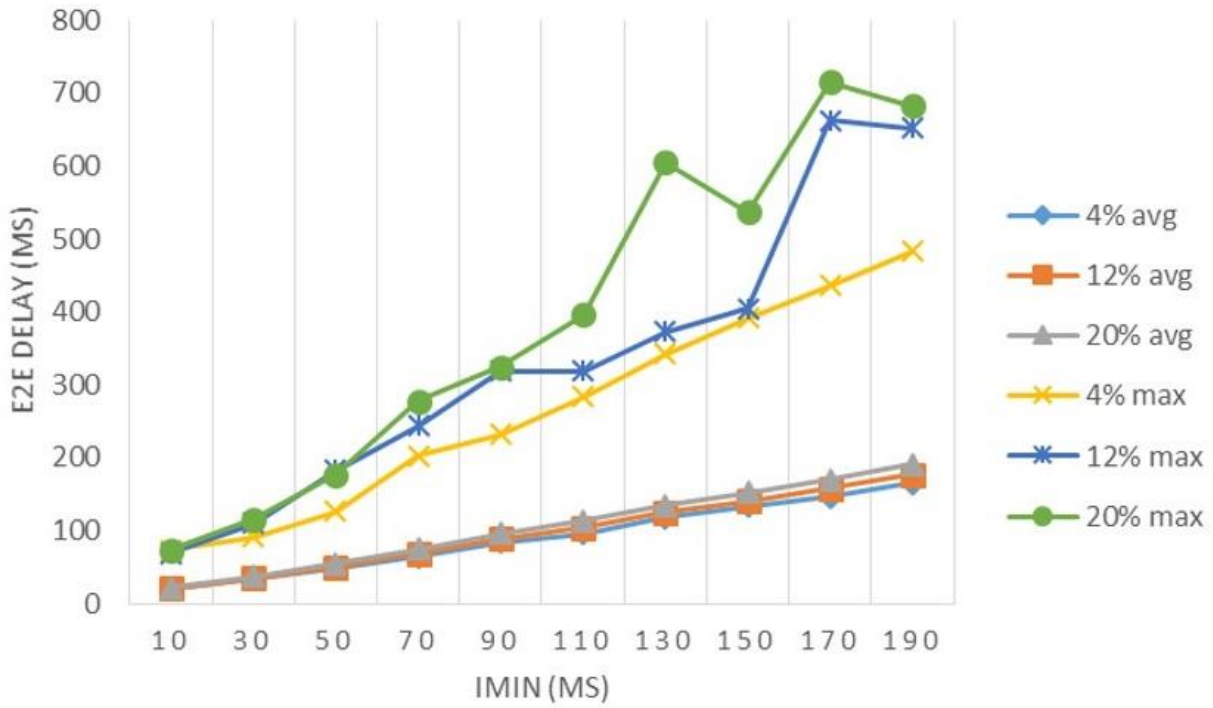

Figure 28 Average and maximum global end-to-end delay as function of I_{min} (ms) at node 22 for 3 transmission loss values

Figure 28 shows the maximum and average global end-to-end delay (ms) at node 22 as function of I_{min} (ms) and for 3 transmission loss values (4%, 12%, and 20%). Figure 28 shows an increase of the average global end-to-end delay (x%-avg curves) with increasing I_{min} value, caused by increasing maximum global end-to-end delay (x%-max curves) and an increasing minimum end-to-end delay after 1 hop. A higher transmission loss probability leads to higher maximum global end-to-end delays, because there exists a higher probability of loss of packets X1 in the forwarders.

9.4.3. Local multicast

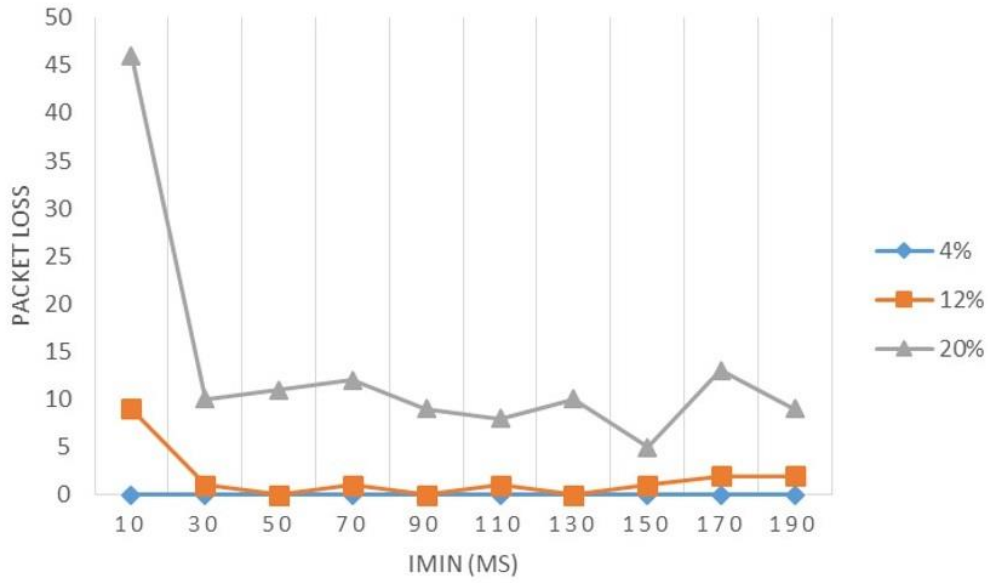

Figure 29, Number of local lost packets at node 22 as function of I_{min} for 3 transmission loss probabilities

Figure 29 shows the losses at node 22 for the packets sent locally by node 21. The increased loss for I_{min} < 20 is caused by the throwing away of S1, S2, X1, and X2 packets due to overload. The loss of 45 packets amounts to a quart permil.

Figure 30 shows the corresponding local end-to-end delay at node 22. The average increases due to an increasing maximum local end-to-end delay. The maximum end-to-end delay comes from packets X2, which increases fast with I_{min}. The contribution of the X2 packets is small, which shows up in the slow increase of the average delays.

The best setting of I_{min} is between 10 ms and 70 ms, leading to maximum global and local end-to-end delays which are smaller than 200 ms and losses which are less than 0.05% given a transmission loss probability of 20%.

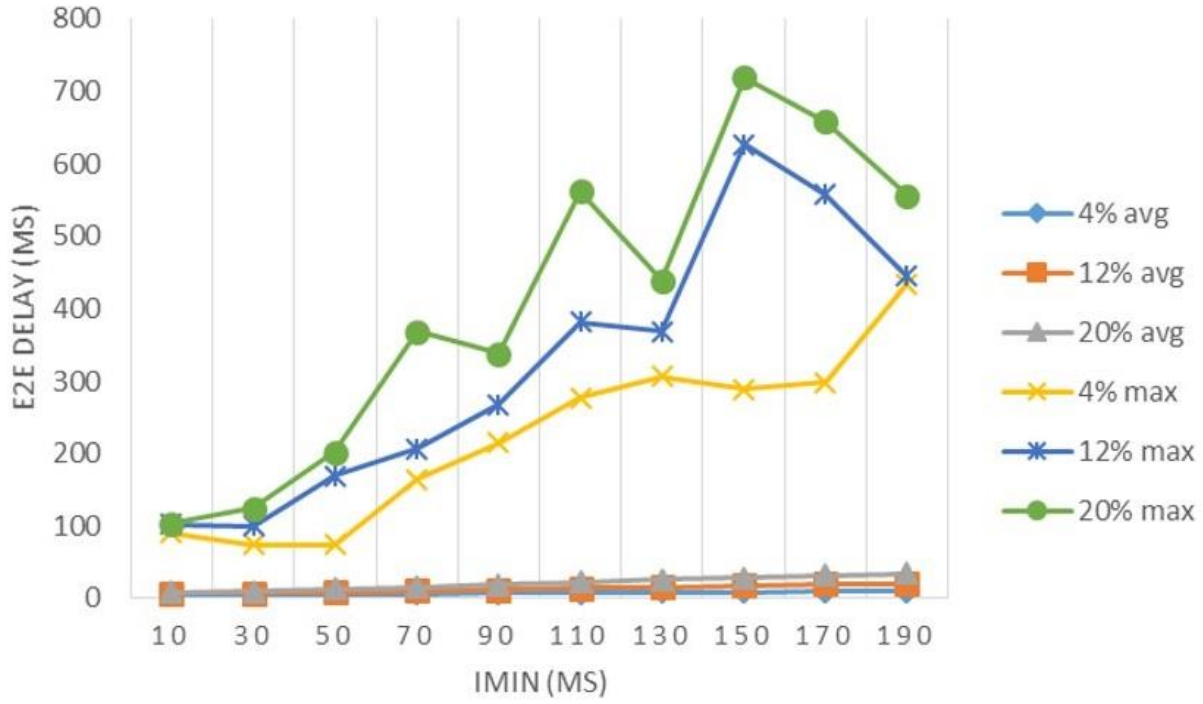

Figure 30, average and maximum local end-to-end delay at node 22 as function of I_{min}

9.4.4. Global local multicast with errors

The simulations show with an increasing communication error the importance of packets X2 increases. The additional use of packets X2 leads to larger end-to-end delays increasing with I_{min}. For I_{min} <30 ms, the local overload leads to additional losses.

10. Lessons learnt

From the former sections a number of lessons is learnt. They are summarized in this section, explaining the recommended setting and the accompanying reasons. Although simulations are done on one type of mesh running 1 global multicast and 5 local multicasts, the lessons are thought to be general enough for many other network topologies.

- **Real-time scheduling** (section 9.1.3)
  - *Reason:* Real-time scheduling favours packets which are close to their deadline and removes packet which are too late anyway.
  - *Setting*: Use of RT_5 algorithm is recommended.
- **Forwarder choice** (section 9.2.3)

  - *Reason:* a point to point transmission loss probability of 20% can provoke too large end-to-end packet losses.
  - *Setting in 3x3 mesh*: a forwarder at the seed and a second forwarder limit losses to 4 permil and end-to-end delays to 200 ms.
  - *Setting in 3x15 mesh*: 3 forwarder pairs limit losses to .5 permil and end-to-end delays to 200 ms.
- **I_{min} and k choice** (section 9.3.3)
  - *Reason:* seed and forwarders increase c value, leading to losses when c > k. With every hop the end-to-end delay increases with increasing I_{min}. Large losses of packets X1 and X2 occur for I_{min} <= 10 due to temporary overload.
  - *Setting:* Choose k > 3, to avoid losses. Choose 10 < I_{min} < 40 to have acceptable end-to-end delays.
- **Forwarder domain** (section 9.4.1)
  - *Reason*: Increasing the number of forwarders increases the network load.
  - *Setting***:** Forwarders only repeat packets for which they are also receivers.
- **Minimum I_{min}** (section 9.4.3)
  - *Reason*: With simultaneous local and global multicasts, small I_{min} values lead to temporary network overload.
  - *Setting*: I_{min} > 10 to avoid packet loss.
- **Communication losses** (section 9.4.4)
  - *Reason*: Due to packet losses the importance of repeat 2 and repeat 1 packets increases.
  - *Setting***:** 10 < I_{min} < 40.

Taking into account that communication losses occur but remain limited to 20%, and assuming that multiple local multicasts coexist with a global multicast, the following **recommendations** are done:

Recommendations

- Two forwarders for 1-hop mesh; two or more forwarders per hop for n-hop mesh
- 10 ms < I_{min} < 40 ms.
- k = 4
- Max_repeat = 2
- Forwarders only active for their domain
- Use RT_5 real-time scheduling algorithm

11. Acknowledgements

The simulation has been made possible by Marc Aoun, who designed a large part of the communication model. I am grateful for discussions with Esko Dijk, Michael Verschoor, Dee Denteneer, Michael van Hartskamp, Armand Lelkens, Jonathan Hui, Richard Kelsey, Don Sturek, and Kerry Lynn.

12. Conclusion

This note describes an analysis of multicast performance on a wireless network with the aid of analytic formulas and simulation results. Given the wide applicability of the analytic results, the major part of the conclusions are valid over a wide range of network topologies. However, the detailed numbers on end-to-end delay and loss probability delivered by the simulations need confirmation by experiments on real networks.

The analysis of the performance of the MPL multicast algorithm indicates that for a large set of installations the maximum end-to-end delays remain within 200 ms and lighting command losses are as low as 1 permil when point to point communication losses of 20% take place. The results recommend parameter settings which are valid over a large range of operational settings

13. References

[1] J. Hui, R. Kelsey, *Multicast Protocol for Low power and lossy networks*, draft-ietf-roll-trickle-macst-07, work in progress, February 2014.

[2] Philip Levis, Neil Patel, Scott Shenker and David Culler, *Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks*, UCB//CSD-03-1290, EECS Department, University of California, Berkeley, CA 94720, 2004.

[3] Jérôme Rousselot, Jean-Dominique Decotignie, Marc Aoun, Peter van der Stok, Ramon Serna Oliver, Gerhard Fohler, *Accurate Timeliness Simulations for Real-TimeWireless Sensor Networks*, Third UKSim European Symposium on Computer Modeling and Simulation, 2009.

[4] IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Appendix A, 2011.

[5] T. He, J.A. Stankovic, C. Lu, T. Abdelzaher, *"SPEED: a stateless protocol for real-time communication in sensor networks,* ICDCS-23, 2003.

[6] M. Aoun, P. van der Stok, Overloading an IEEE 802.15.4 Point-to-Point connection with Real-Time Messages. RTSS 2010:225-235.