Intl. Conf. in Computer Sciences - RIVF'06
February 12-16, 2006, Hochiminh City, Vietnam

The impact of the implosion problem on the backoff time for a reliable multicast protocol

Ngo Le Minh, Alain Jean-Marie

minhnl@vnu.edu.vn, ajm@lirrm.fr

Abstract **This paper studies the implosion problem in multicast. The result explained in this paper is a part of the work on analyzing the scalability of a reliable multicast protocol. The paper describes a method proposed to determine the backoff time for multicast clients to avoid packet loss due to the implosion problem. The proposed method has been implemented and the output data is shown. The flow of acknowledgment packets from clients to the server has been studied. Random multicast networks are generated and backoff times are calculated in order to show the impact of the implosion problem on the transmission delay in reliable multicast. It is proved by analysis that the backoff interval is proportional to the number of users which is large enough. Linear increase in transmission delay is equivalent to that a reliable multicast protocol fails to work for a large number of users.**

Index Terms**— reliable multicast, implosion problem, backoff time, performance analysis, scalability analysis**

1. Introduction

Implosion problem is well known among designers of reliable multicast protocol. The implosion problem occurs when a large number of acknowledgment (ACK) packets generated by clients are sent simultaneously to a server. The delay is impacted the most by this problem. A large number of reliable protocols have been proposed [1] [2] [3] [4] [5] [6] in the past years, each protocol has its owned method to miniature the problem. However none of the works has been done to qualitatively analyze the problem. This paper aims to find out the exact time interval the server needs to get all ACK packets sent out by a specific number of users. The demand for discussing this matter appears during the investigation of the scalability of a reliable multicast protocol proposed for a unidirectional satellite link. The protocol is described in details in [7]. It seems that when the number of users increases the time interval the server must wait for clients to send ACK packets increases and mainly because of the increase in the backoff time. The flow of ACK packets heading to the server is explored to figure out the dependency of the backoff time on the number of users.

The paper is organized as follows. Session 1 is an introduction. Session 2 describes the context in which the study is carried out. Session 3 presents a proposed algorithm to find a time interval called calculated backoff. Calculated backoff is analyzed and it is found out that calculated backoff is not always appropriate for setting the backoff time. Calculated backoff is replaced by minimum backoff which is explained in session 4. Session 5 describes mathematical formulas created to verify the result of session 3 and 4. Session 6 shows simulation data. Session 7 studies randomly generated networks. Session 8 is the conclusion.

2. The context for the study

According to the protocol described in [7], every client generates a response packet and sends to the server whenever it receives a request packet from the server. Request packet can be one of the following types: RQ (Query Request) for collecting client information, ME (Monitor End) for congestion control, POLL_ALL/POLL_INDV for querying information about missing packets. Response packets can be one of the following types: RJ (Receiver Join) in response to RQ, the length is 63 bytes, MR (Monitor Report) to ME, 63 bytes, RR (Reception Report) to POLL_ALL/POLL_INDV, its length is 170 bytes for 2.5 Mb transmission file. Data packets and control packets from the server are multicast to client via satellite downstream links. Upstream links either do not exist or have a limited capacity so that clients must send ACK packets over the Internet through the support of Link Layer Tunnelling Mechanism [8] installed at the link layer. The physical configuration of the protocol is showed in the figure below:

Ngo Le Minh is with College of Technology, National University in Hanoi, 144 Xuanthuy, Caugiay, Hanoi, Vietnam
Alain Jean-Marie is with LIRMM, 161 Rue Ada, 34392 Montpellier Cedex 05, France

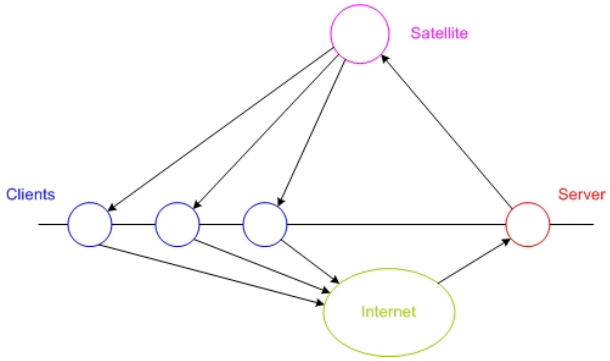

Fig. 1 The physical configuration for communication

For the purpose of evaluating the performance of the protocol, the need of finding out the time interval for clients to back off arises. The investigation is crucial as the backoff time has an influence on the transmission delay and is impacted the number of participants. For the purpose of analyzing the scalability of the protocol, it is important to discover the dependency between the number of users and the backoff time. The next sessions will clarify these issues.

3. An algorithm for determining calculated backoff

It is well known that packets are forwarded by routers in the Internet. When they arrive at the buffer of a router in an amount bigger than the size of the router buffer the buffer is overflowed and some packets are dropped. If the overflow happens due to many clients generate packets and send them immediately as in multicast, the only way to avoid it is allowing clients backing off before sending a packet. All packets originated from multicast clients must go through the first-hop router of the server and be forwarded by the router to the final destination. And this is the location where response packets can be dropped due to buffer overflow.

To deal with the overflow problem, packets must somehow arrive at the buffer regularly after a time interval equal the transmission delay of the previous packet and be freed with being queued. It is absolutely possible if each client backs off randomly using uniform distribution over the time interval equal the transmission delay of the number of packets generated by clients. This measure would cause packets to arrive regularly at the buffer and thus solve the mass dropping problem due to buffer overflow. In general, if there are t packets to come to the buffer then the backoff interval is t * the transmission delay of a packet or equals t * packet length / link bandwidth.

In a multicast session, users may reside in various networks. Assume that clients locate in several networks that have various latencies and different numbers of users. This assumption is depicted in the figure below.

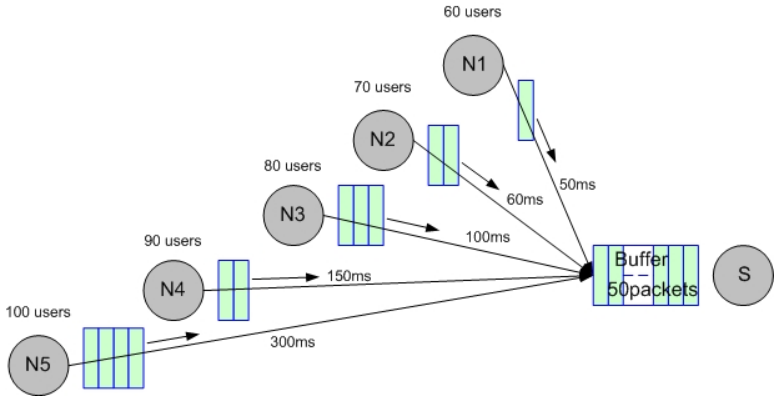

Fig. 2: Networks in a multicast session

Fig. 2 shows an example. There are 5 networks {N1, N2, N3, N4, N5}. The latencies of networks to the server gateway respectively are {50ms, 60ms, 100ms, 150ms, 300ms}. The numbers of users are {60, 70, 80, 90, 100} users in {N1, N2, N3, N4, N5}. Control packets generated by users in network N1 arrive to the server buffer earlier then those of network N2 as N1 has shorter latency than N2. Here, a question occurs. What is the backoff time set for this configuration such that packets generated by clients do not cause buffer overflow at the server gateway?

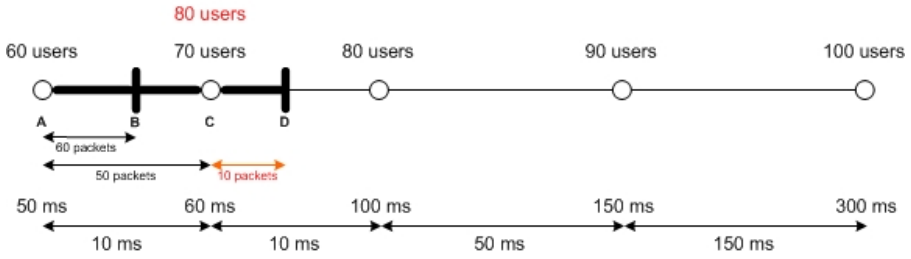

Fig. 3 Networks with ascending order of latencies

Fig. 3 is another way of representing Fig. 2. Little circles depict networks. Networks are arranged in ascending order of latencies. Above and below each circle is the number of users in the network denoted by the circle and its latency. The first network has 60 users and 50ms latency. The difference in latency between N1 and N2 is 10ms. 60 users in N1 generate 60 control packets at the time they get a request packet multicast from the server. 60 packets from N1 would arrive at the server buffer at the same time, thus the backoff interval for N1 is the transmission delay of 60 packets. If we assume that the difference in latency between N1 and N2 depicted by AC is bigger than the backoff interval for N1 depicted by AB, then packets from N1 will arrive at the gateway buffer S before any packet from N2. In this case, packets generated by N1 do not interfere with packets generated by N2. Now we assume that the difference in latency between N1 and N2 is less than the backoff interval for N1 depicted by AD. For example, the time interval AC is the backoff time of 50 packets. Thus 10 packets would arrive at the gateway buffer S at the same time with packets from N2. Therefore these 10 packets must be added to 70 packets of N2, so that their transmission delay is taken into account in computing the backoff time for N2. The backoff interval for N2 now equals the transmission delay of 80 packets, not 70 packets as initially. This adding up ensures that 10 packets of N1 do not cause any of 70 packets of N2 to be dropped due to buffer overflow. Now the number of users of N2 becomes 80, repeat again the comparison between N2 and N3 to find out what is the number of backoff packets for N3. And continue until the last network to find out the number of backoff packets for each network. The largest number of backoff packets among networks is taken to compute the backoff interval for the whole multicast network. This backoff interval is called calculated backoff.

Given three factors, the number of networks, the number of users in each network, the latency and the size of the control packet, a program is implemented to find calculated backoff for any multicast network.

Now we introduce the configuration of experimental networks in order to illustrate the algorithm and also use them for the next sessions. The following set of users is chosen {100, 500, 1000, 2000, 3000, 5000, 8000, 10000}. Each network has 100 users. Accordingly, the number of networks corresponding to each group of users is the following set {1, 5, 10, 20, 30, 50, 80, 100}. The maximum latency is 300ms. Two neighbour networks have the difference in latency equal 200ms / the number of networks. The following set denotes the distance in latency between two neighbour networks {N/A, 40ms, 20ms, 10ms, 7ms, 4ms, 2.5ms, 2ms}. Link capacity is 1.5Mbps. The network configuration settings are summarized in the table below:

No of users	**100**	**500**	**1000**	**2000**	**3000**	**5000**	**8000**	**10000**
No of networks	1	5	10	20	30	50	80	100
Difference in latency		0.04s	0.02s	0.01s	0.007s	0.004s	0.0025s	0.002s

Table I Network configuration settings

The example configuration for 500 users is drawn as below:

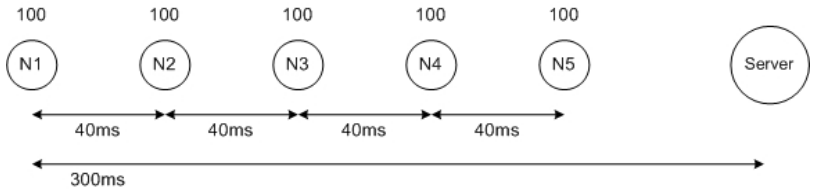

Fig. 4 Network configuration setting for 500 users

We apply the algorithm explained above to determine calculated backoff for 3 types of ACK packets, RJ, MR, and RR. As mentioned before, the size of RJ and MR is 63 bytes, RR – 170 bytes. Calculated backoffs for experimental networks are given in Table II. The table also contains big backoff which is simply calculated by the total number of users * the transmission delay of the ACK packet.

No of users		**100**	**500**	**1000**	**2000**	**3000**	**5000**	**8000**	**10000**
RJ,	**Calculated backoff**	0.04s	0.04s	0.16s	0.49s	0.85s	1.5s	2.51s	3.2s

MR	**Big Backoff**	0.04s	0.17s	0.34s	0.68s	1.01s	1.68s	2.67s	3.36s
RR	**Calculated backoff**	0.1s	0.3s	0.73s	1.63s	2.54s	4.36s	7.12s	8.89s
	Big Backoff	0.1s	0.46s	0.91s	1.82s	2.72s	4.54s	7.26s	9.07s

Table II Calculated backoff and big backoff

It is shown in Table II that calculated backoff is bigger for a bigger group. For 10000 users it becomes quite large value. Calculated backoff is always less than big backoff.

4. Determining minimum backoff

There is a question that is there any possibility that calculated backoff still leads to buffer overflow at the server gateway? The backoff rule using calculated backoff is illustrated in the figure below:

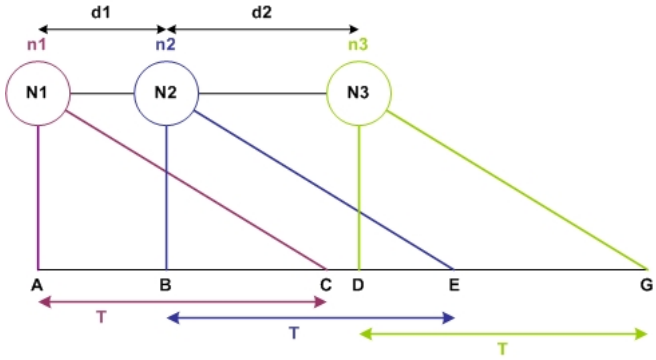

Fig. 5 Calculated backoff for three networks

In Fig. 5, there are three networks denoted by {N1, N2, N3}. The number of users in each network is {n1, n2, n3}. The difference in latency between N1 and N2, N2 and N3 is {d1, d2}. The calculated backoff is denoted by T. AC, BE and DG are the intervals during which packets from N1, N2, N3 are supposed to arrive at the server gateway. BC and DE are overlapping intervals during which arriving packets come from two networks N1, N2 and N2, N3 respectively. If there are no overlapping intervals then it is guaranteed that the buffer is not overflowed. But the arrival rate increases if there is more than one network generating ACK packets. Denote t_p the transmission delay of a packet, the maximum link throughput of the server gateway is $\frac{1}{t_p}$. We will compare the maximum arrival rate with the maximum link throughput to see whether the maximum arrival rate of packets generated by networks exceeds the maximum link throughput of the server gateway.

The algorithm to find the maximum arrival rate is as follows. Firstly, we find the maximum arrival rate of each network. In Fig. 5, it is the maximum rate over AC for N1, BC for N2 and DG for N3. Because a network generates packets regularly, its rate equal its number of users / T, the arrival rate of ACKs for AB is the arrival rate of N1, but BC has the arrival rate equal the sum of the arrival rates of N1 and N2. BC of course has higher arrival rate than AB. Because T is unchanged for all networks, it is quite easy to get the maximum arrival rate of a network. Add T to its latency. As a result, all the networks that are on the left of it and have latency less than the sum will have their backoff intervals overlapping the backoff interval of the network itself. Adding up the rates of all found networks to the rate of the network itself gives the maximum rate of it. The last step is choosing the maximum arrival rate among all networks.

The algorithm has been implemented by a program. The maximum rates of ACKs for experimental networks are computed and showed in the next table. The unit of rate is packets per second.

No of users		**100**	**500**	**1000**	**2000**	**3000**	**5000**	**8000**	**10000**
RJ, MR	**Calculated backoff**	0.04s	0.04s	0.16s	0.49s	0.85s	1.5s	2.51s	3.2s
	Big Backoff	0.04s	0.17s	0.34s	0.68s	1.01s	1.68s	2.67s	3.36s
	Max rate of ACKs	2500	2500	5000	4082	3529	3334	3188	3126
	Max throughput	2977							
RR	**Calculated backoff**	0.1s	0.3s	0.73s	1.63s	2.54s	4.36s	7.12s	8.89s
	Big Backoff	0.1s	0.46s	0.91s	1.82s	2.72s	4.54s	7.26s	9.07s
	Max rate of ACKs	1000	1667	1370	1227	1182	1147	1124	1125
	Max throughput	1103							

Table III Maximum rate of ACKs

Data in Table III shows that most cases manifest the maximum arrival rate higher than the maximum link throughput of the server gateway. This means there is very high probability that the server buffer is built up and packets are dropped when the buffer becomes full.

If the maximum arrival rate of ACKs is higher then the maximum throughput of the server gateway, the calculated backoff interval should be enlarged to reduce the rate. The enlargement must be repeated until the maximum arrival rate of ACKs is equal or higher than the maximum throughput. At this point, the calculated backoff reaches some value called minimum backoff. This is the backoff interval we need because it limits the maximum rate of ACKs to a value less than the maximum throughput of the server gateway. The following table contains maximum backoff intervals computed for experimental networks:

No of users		**100**	**500**	**1000**	**2000**	**3000**	**5000**	**8000**	**10000**
RJ, MR	**Calculated backoff**	0.04s	0.04s	0.16s	0.49s	0.85s	1.5s	2.51s	3.2s
	Big Backoff	0.04s	0.17s	0.34s	0.68s	1.01s	1.68s	2.67s	3.36s
	Max rate of ACKs (pkts)	2500	2500	5000	4082	3529	3334	3188	3126
	Max throughput (pkts)	2977							
	Minimum Backoff			0.34s	0.68s	1.01s	1.68s	2.67s	3.36s
RR	**Calculated backoff**	0.1s	0.3s	0.73s	1.63s	2.54s	4.36s	7.12s	8.89s
	Big Backoff	0.1s	0.46s	0.91s	1.82s	2.72s	4.54s	7.26s	9.07s
	Max rate of ACKs (pkts)	1000	1667	1370	1227	1182	1147	1124	1125
	Max throughput (pkts)	1103							
	Minimum Backoff		0.46s	0.91s	1.82s	2.72s	4.54s	7.26s	9.07s

Table IV Minimum backoff

Table IV shows the fact that for most networks the minimum backoff becomes exactly the same as the big backoff of that network.

5. Mathematical analysis of backoff for experimental networks

In this session, we work out mathematical formulas to compute calculated backoffs and maximum arrival rates of ACKs for experimental networks in Table I. Their settings are redrawn in the figure below:

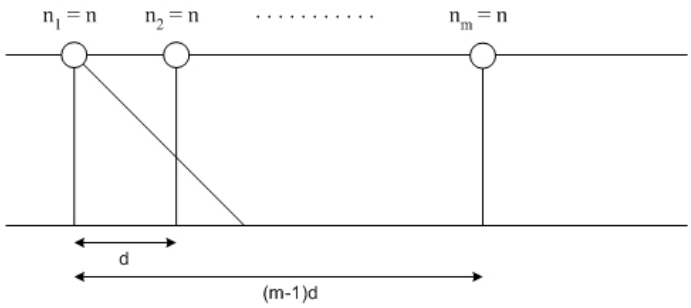

Fig. 6 Configuration settings for experimental networks

In Fig. 6: m = the number of networks; n = the population of each network; m x n = the total population; d = the difference in latency between two succession networks; m x d = 0.2s

Denote: t_p: the transmission delay of an ACK packet; T: calculated backoff; R_{max}: Maximum rate of ACKs

The backoff time T is computed by the following formula:

If $n \times t_p \le d \rightarrow \quad T = n \times t_p$ Formula 5.1

If $n \times t_p > d \rightarrow \quad T = n \times m \times t_p - (m-1) \times d$ Formula 5.2

In the second case, every network has some packets that are added up to its neighbouring network, the maximum number of backoff packets is found in the last network. $(m-1) \times d$ is the time interval that packets are not added up. Hence, the calculated backoff T is computed by Formula 5.2.

Maximum arrival rate of ACKs R_{max} is computed as below:

If $n \times t_p \le d \rightarrow \quad R_{\max} = n/T$

Substitute T in formula 5.1:

$$R_{\max} = n/(n \times t_p) = \frac{1}{t_p} \qquad \text{Formula 4.3}$$

If $n \times t_p > d$ there are two cases:

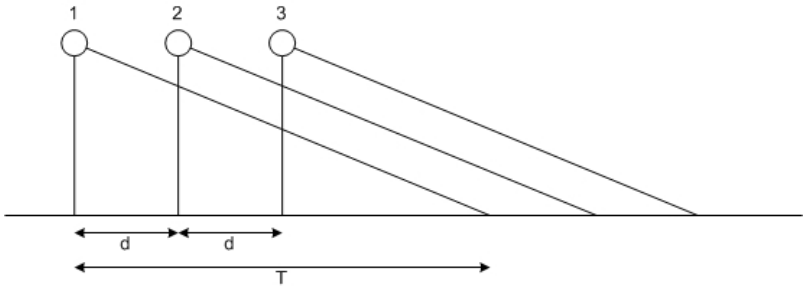

Fig. 7 The first case for computing maximum arrival rate of ACKs

The first case takes place when $m \le ceiling(\frac{T}{d})$. The rates of ACK in consecutive intervals starting from network 1 in figure 4.9 is $\{\frac{n}{T}, \frac{2n}{T}, \frac{3n}{T}, \frac{2n}{T}, \frac{n}{T}\}$. The formula for R_{max} is:

$$R_{\max} = m \times (n/T) \qquad \text{Formula 4.4}$$

The second case is depicted in the figure below:

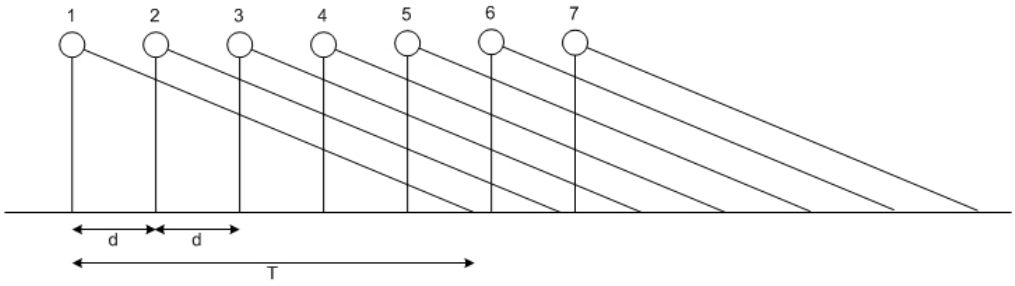

Fig 8 The second case in computing maximum rate of ACKs

The second case is when $m > ceiling(\frac{T}{d})$. The consecutive rates of ACKs are $\frac{n}{T} \times \{1, 2, 3, 4, 5, 4, 5, 4, 5, 4, 3, 2, 1\}$. R_{max} is this case is:

$$R_{\max} = ceiling(\frac{T}{d}) \times (\frac{n}{T}) \qquad \text{Formula 4.5}$$

Combining two cases, the formula for R_{max} is

$$R_{\max} = \min(m, ceiling(\frac{T}{d})) \times (\frac{n}{T}) \qquad \text{Formula 4.6}$$

Now we apply derived formulas to verify calculated backoffs and maximum rates of ACKs that are computed programmatically.

	100	**500**	**1000**	**2000**
N	100			
M	1	5	10	20

D	∞	0.04s	0.02s	0.01s
t_p	$63*8/(1.5*10^6) = 0.000336$s			
$n*t_p$	$0.0336 \approx 0.04$s			
T	**0.04s**	**0.04s**	= 1000*0.000336 – (10-1)*0.02 = 0.336 – 0.18 = 0.156 ≈ **0.16s**	= 2000*0.000336 – (20-1)*0.01 = 0.672 – 0.19 = 0.482 ≈ **0.49s**
Ceiling(T/d)			= ceiling(0.16/0.02) = 8	= ceiling(0.49/0.01) = 49
R_{max}	= 100/0.04 = **2500**	= 100/0.04 = **2500**	= 8*(100/0.16) = **5000**	= 20*(100/0.49) = **4082**

Table V The computation of calculated backoff and maximum arrival rate of ACKs

The result drawn programmatically and shown in Table III coincides with the result computed using mathematical formulas.

6. Analysis on the queue build-up

Having the method to compute minimum backoff, we are going simulate the flow of ACK packets travelling from clients to the server and observer the buffer of the server gateway to witness the build-up of the queue. A number of simulations have been implemented in NS-2. The simulation scenario is drawn in Fig. 9. Clients in all networks denoted by small circles on the left in Fig. 9 back off randomly over the minimum backoff interval, after backing off each client sends a packet to the server which is denoted by the big circle with the label "Server" on the right. Packets from clients converge to the server gateway denoted by the small circle with the label "R". The buffer of the link from R to Server is traced. Trace data created after each simulation is stored in a file that later is analyzed to gather the maximum number of packets queued in the buffer and the number of ACK packets received by the server. No other traffic than from multicast clients is injected.

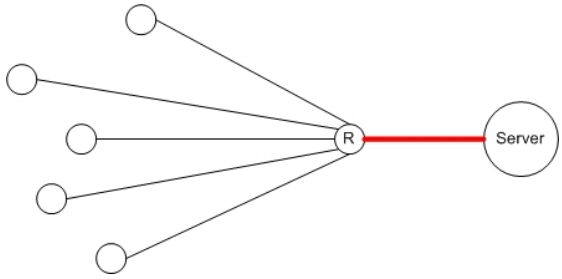

Fig. 9 The scenario for the investigation of the buffer build-up

Simulations are carried out for all experimental networks. Analyzed data on the maximum size of the queue, the number of packets arrived at the server and the percentage of packets that are dropped due to buffer overflow is showed in the table below:

		100	**500**	**1000**	**2000**	**3000**	**5000**	**8000**	**10000**
Calculated backoff	**Max queue**	1	1	OF	OF	OF	OF	OF	OF
	No. Pkts received	100	500	824	1639	2660	4597	7566	9597
	% dropped	0%	0%	17.6%	18.1%	11.4%	8.1%	5.5%	4.1%
Minimum backoff	**Max queue**			25	32	43	OF	OF	OF
	No Pkts received			1000	2000	3000	4984	7954	9948
	% dropped			0%	0%	0%	0.32%	0.58%	0.52%

Table VI Analyzed data on buffer overflow and packet loss

OF: Overflow

Data in Table VI shows the buffer build-up and the amount of lost packets. The percentage of lost packets is very high for calculated backoff. If clients back off longer, using minimum backoff, the buffer is still built up, overflowed in some cases, but the number of lost packets reduces significantly, less than 1% for all networks. The reduction of packet loss means that clients must increase backoff time to reduce the loss. Minimum backoff is proved to be long enough since it does not significantly induce packet loss.

Now we inject some traffic into the server gateway to investigate what happens to the queue. In order to do so, simulation scenario remains the same. The CBR traffic with the rate occupied 20% of the link capacity is injected to the gateway node. With the additional traffic, the buffer is exposed as follows:

		100	500	1000	2000	3000	5000	8000	10000
Minimum backoff	**Max queue**	9	19	OF	OF	OF	OF	OF	OF
	No. Pkts received	100	500	948	1810	2617	4276	6728	8418
	% dropped	0%	0%	5.2%	9.5%	12.8%	14.5%	15.9%	15.9%

Table VII The buffer with the presence of additional traffic

Packet loss rate again increases as the result of buffer overflow. Clients again have to enlarge backoff interval in order to avoid packet loss. This fickleness leads to more complicated situation when backoff time must be adjusted in accordance with the network condition.

7. Backoff for random networks

Experimental networks, in fact, result in large minimum backoffs, as large as big backoffs. We want to know the average ratio of minimum backoff to big backoff. If the ration is less than 1, the minimum backoff is shorter than the big backoff, the proposed method is of benefit in shortening the backoff time. If the ratio is close to 1, the minimum backoff is the same as the big backoff. For this case, the big backoff is of benefit for its simple rule. The matter should be investigated on a number of networks randomly generated. The minimum backoff, the big backoff and their ratio are to be computed for each network.

The same set of the total number of users {100, 500, 1000, 2000, 3000, 5000, 8000, 10000} is used for investigation. For every number of users, a number of networks have been tested and are the set {1, 5, 8, 10, 20, 30, 50, 80, 100}. The number of users in each network is randomly picked up. The latency of each network is chosen randomly from 20ms up to 500ms. The link capacity is 1.5Mbps. Clients send MR packets. Networks are generated by a program. An example of a generated network is printed below. The example multicast network has 1000 users located in 5 networks. Networks are sorted in ascending order of latency.

```
NumberOfNets: 5 Test 1
1 users: 325 latency: 46.0
2 users: 17 latency: 141.0
3 users: 73 latency: 156.0
4 users: 284 latency: 387.0
5 users: 301 latency: 467.0
Backoff(packets):347 CalculatedBackoff:0.116592 MaxRate:5018.0
MinimumBackoff:0.196592 BigBackoff:0.336 Ratio: 0.585095238095238
```

For each group of users, a large number of networks are generated; the ratio of the minimum backoff to the big backoff is calculated. Then the ratio is categorized based on its value. The ratio which is in the range [0, 0.1) belongs to group 1, group 2 has the range [0.1, 0.2) etc. After that, the ratio is averaged. Statistical data is given in Table VIII

		100		500		1000		2000	
Total No of tests		250		400		400		400	
		No of tests	**%**	**No of tests**	**%**	**No of tests**	**%**	**No of tests**	**%**
0	**0.1**	1	0%	3	0%	0	0%	0	0%
0.1	**0.2**	70	28%	126	31%	3	0%	0	0%
0.2	**0.3**	109	43%	110	27%	58	14%	0	0%
0.3	**0.4**	54	21%	63	15%	82	20%	0	0%
0.4	**0.5**	12	4%	40	10%	76	19%	0	0%
0.5	**0.6**	0	0%	20	5%	49	12%	0	0%
0.6	**0.7**	3	1%	16	4%	39	9%	0	0%
0.7	**0.8**	1	0%	11	2%	31	7%	0	0%
0.8	**0.9**	0	0%	6	1%	24	6%	0	0%
0.9	**1.0**	0	0%	4	1%	16	4%	60	15%

1.0		0	0%	1	0%	22	5%	340	85%
Average ratio		0.257		0.318		0.528		1.000	
		3000		**5000**		**8000**		**10000**	
Total No of tests		400		400		400		400	
		No of tests	**%**	**No of tests**	**%**	**No of tests**	**%**	**No of tests**	**%**
0	**0.1**	0	0%	0	0%	0	0%	0	0%
0.1	**0.2**	0	0%	0	0%	0	0%	0	0%
0.2	**0.3**	0	0%	0	0%	0	0%	0	0%
0.3	**0.4**	0	0%	0	0%	0	0%	0	0%
0.4	**0.5**	0	0%	0	0%	0	0%	0	0%
0.5	**0.6**	0	0%	0	0%	0	0%	0	0%
0.6	**0.7**	0	0%	0	0%	0	0%	0	0%
0.7	**0.8**	0	0%	0	0%	0	0%	0	0%
0.8	**0.9**	0	0%	0	0%	0	0%	0	0%
0.9	**1.0**	113	28%	180	45%	291	72%	357	89%
1.0		287	71%	220	55%	109	27%	43	10%
Average ratio		1.000		1.000		0.9999		0.9998	

Table VIII Statistical data for randomly generated networks

The distribution of ratio is shown in Fig. 10. The average ratio is illustrated in Fig. 11.

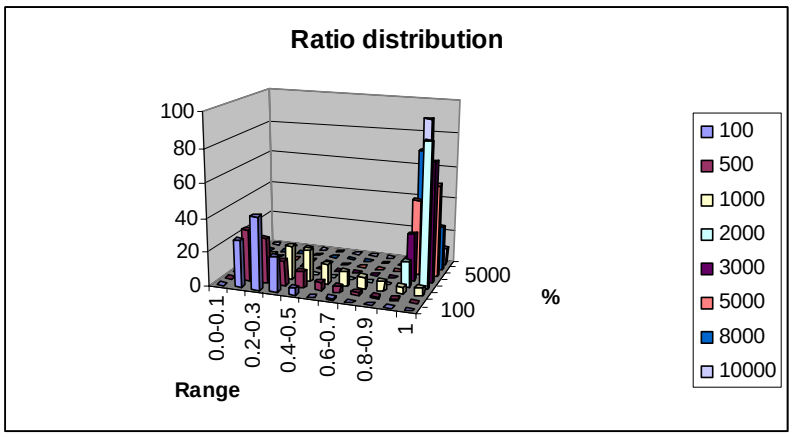

Fig. 10 Ratio distribution

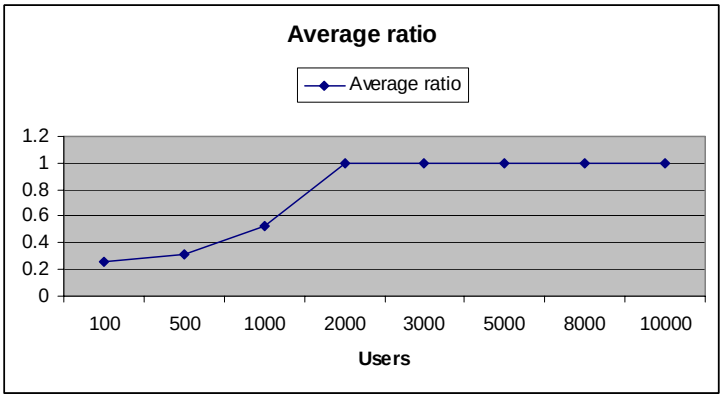

Fig. 11 Average ratio

Observing Fig. 10 and analyzing data in Table VIII lead to the point that when users are few, the distribution of ratio is close to 0. But with the growth of the number of users the distribution gradually moves toward 1. Fig. 11 shows that the backoff time is proportional to the number of users when the ratio increases up to 1, or when the minimum backoff is the same as the big backoff.

Generated statistical data shows that backoff time increases linearly with the number of users. The increase in backoff time of a multicast protocol results in the increase in delay or the decrease in throughput. The implosion problem does influence on the performance of reliable multicast protocols and must be solved to be able to handle any number of users.

8. Conclusions

The paper has explained a method to compute the backoff time based on the number of users and the predefined configuration of a multicast network. This method should be applied only for the purpose of studying the performance of a protocol assumed that all the parameters for the computation are known, not for using it in the Internet where parameters, such as topology, bandwidth capacity and latency are unknown. Moreover, the method does not dynamically adapt to the fluctuation of the Internet.

The paper has pointed out that the backoff time depends on the number of users and increases linearly with it. Thus the transmission delay grows or the performance goes down with the rise of the number of users. To be scalable, a reliable multicast protocol must be designed to limit the number of ACK packets simultaneously sent by multicast clients to a single point, the server.

Future work on finding a more suitable method for the Internet has to be discussed. Congestion control and handling implosion problem in reliable multicast are also issues to be worked out.

Abbreviations

ACK	Acknowledgement
CBR	Constant Bit Rate
ME	Monitor End
MR	Monitor Report
RJ	Monitor Join
RQ	Query Request
RR	Reception Report
POLL_ALL	Polling All
POLL_INDV	Polling Individual

References

[1] S. Floyd, V. Jacobson, S. McCanne, *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing*, IEEE/ACM Transactions on Networking, Vol. 5, No. 6, 1997

[2] L. Rizzo, L. Vicisano, *A Reliable Multicast data Distribution Protocol based on software FEC techniques*, The Fourth IEEE Workshop on the Architecture and Implementation of High Performance Communication Systems, Sani Beach, Chalkidiki, Greece, 1997

[3] C. Papadopoulos, G. Parulkar, G. Varghese, *An error control scheme for large-scale multicast applications*, The Conference on Computer Communications (IEEE Infocom), San Francisco, California, 1998

[4] B. Whetten, M. Basavaiah, S. PaulT. Montgomery, N. Rastogi, J. Conlan, T. Yeh, *THE RMTP-II PROTOCOL*, Internet Draft: RMTP-II Specification, 1998

[5] Kenneth P. Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu, Yaron Minsky, *Bimodal Multicast*, ACM Transactions on Computer Systems, Vol. 17, No. 2, 1999

[6] Sneha Kumar Kasera, Gísli Hjálmtýsson, Donald F. Towsley, James F. Kurose, *Scalable reliable multicast using multiple multicast channels*, IEEE/ACM Transactions on Networking, Vol. 8, No. 3, 2000

[7] P. Basu, K. Kanchanasut, *A Reliable Multicast Protocol for Unidirectional Satellite Link*, Symposium on Applications and the Internet, 2003

[8] E. Duros, W. Dabbous, H. Izumiyama, N. Fjuii and Y.Zhang, *A Link Layer Tunnelling Mechanism for Unidirectional Links*, RFC 3077, 2001