Is the Zero-Wait Policy Always Optimum for Information Freshness (Peak Age) or Throughput?

Basel Barakat, Simeon Keates, Ian Wassell and Kamran Arshad.

***Abstract*—The Zero-Wait (ZW) policy is widely held to achieve maximum information 'freshness', i.e., to achieve minimum Peak Age (PA) and maximum throughput, for real-time Internet-of-Things applications. In this paper, it was shown through a series of experiments that the ZW policy is not necessarily the optimum policy for freshness nor throughput in all real-world scenarios. Firstly, the effect of delay on the ZW policy was shown on a LAN. Afterward, the server was located on the Internet, and it was shown that the ZW policy incurred a two-fold PA and throughput performance degradation compared with continuously sending status updates.**

***Index Terms*—Real-Time Systems, Status Update, Information Freshness, Peak Age of information, Zero-Wait Policy.**

I. Introduction

ONE of the main challenges for designing Internet of Things (IoT) applications is how to deliver information in a sufficiently t imely m anner. O utdated i nformation may interfere with the operation of critical applications, potentially endangering lives, such as in telehealth or autonomous cars. Recently, a new metric was proposed to measure and quantify the information 'freshness', i.e., Age of Information (AoI) [1]. More recently, the Peak Age (PA) metric, which is defined as the mean maximum AoI of a piece of information, was proposed as a more tangible (utilizable) metric [2], [3]. Both metrics look at the information freshness from the destination's perspective, hence are reactive measures. They do not determine a proactive policy for the sending of the information to ensure maximum freshness. This raises the following question: *"How can we ensure that we are delivering the 'freshest' possible information from a time-varying process?"*.

Several policies have been proposed in the literature to achieve minimum PA, i.e., maximum information 'freshness', such as [1]-[5]. One widely used policy is the Zero-Wait (ZW) policy, which minimizes the PA by eliminating the waiting (queuing) time, as explained in section II. The ZW policy is often referred to as a logical mechanism for minimizing the PA and maximizing the number of communicated status updates per time, i.e., maximizing the throughput [5]. Recently it was shown that the ZW policy is unable to achieve optimum PA performance when the status update service time changes continuously between long and short duration [5]. However, it can be argued that such scenario only reflects one special case and might not be the best characterization of the majority of real-time IoT applications. Nevertheless, the ZW policy is still considered as an optimum throughput policy [5]. Furthermore, most of the published work to minimize PA is purely theoretical without proof-of-concept. Additionally, no previous work investigated the performance of ZW in a cloud scenario (in which the conventional theoretical models fail to evaluate the PA).

In this paper, the outcomes of an experimental study are presented to evaluate the PA and inter-arrival time (INT) of the ZW policy. Two scenarios are proposed, tested and evaluated. The first scenario (S1) has both the source (client) and the destination (server) of the status updates located in the same Local Area Network (LAN). In the second scenario (S2), the server is located in the cloud, i.e., hosted by a remote service provider. Firstly, the effect of delay on the ZW policy was investigated by deriving an expression for the ZW PA performance. Afterward, the threshold delay value in which the ZW fails to deliver the freshest information is presented in section III. The PA and INT performance of the ZW policy are compared with those achieved by the Continuous Updating (CU) policy. The experiment system model is presented in section IV.

The PA and INT performance of the ZW policy was significantly different in the two given scenarios, as shown in section V. In S1, the ZW PA performance clearly outperformed the CU policy, as predicted. However, the ZW PA performance in S2 is much worse than in S1. In particular, the CU policy outperformed the ZW in terms of PA and INT by a factor of two in S2. The results of statistical tests are also presented to validate the results. Section III presents the cases in which the ZW policy fails to achieve the most optimum PA performance. Finally, conclusions and proposals for future work are presented in section VI.

II. Peak Age and Zero-Wait Policy

The PA metric is defined as the mean maximum elapsed time since the latest status update received, was generated [2]. Let I_n be the period between generating two consecutive status updates, i.e., INT. T_n is the status update delay time (T), i.e., the time between the generation and completion of processing update n or $T_n = E\{1/\mu_n + W_n\}$. Hence, P_n is given as [3]

$$P_n = E\{I_n + T_n\} = E\left\{\frac{1}{\lambda_n} + \frac{1}{\mu_n} + W_n\right\}, \forall n. \quad (1)$$

B. Barakat is with the Faculty of Engineering and Science, University of Greenwich, ME4 4TB, Kent, UK. (email:bb141@gre.ac.uk)

S. Keates is with the School of Engineering and the Built Environment, Edinburgh Napier University, Edinburgh, UK. (email:S.Keates@napier.ac.uk)

I. Wassell is with the Computer Laboratory, University of Cambridge, CB3 0FD, UK.(email:ijw24@cam.ac.uk)

K. Arshad is with the Department of Electrical Engineering, Ajman University of Science & Technology, Ajman 346, United Arab Emirates (e-mail: k.arshad@ajman.ac.ae)

Where $E\{.\}$ is the expectation operator, $1/\mu_n$ is the status service time and W_n is the waiting time. In other words, the PA is equal to the inter-arrival duration plus the delay time.

It can be argued that ZW can minimize PA and I_n by achieving a zero-waiting time i.e., $W_n = 0, \forall n$ [3]. In the ZW policy, the clients (e.g., sensors) may generate status updates only when the server is idle, the ZW system model is shown in Fig. 1. Therefore, the clients must wait for the server to send an Acknowledgment (ACK) for each status update [5]. Hence, I_n depends on the delay time of both the status update and the ACK (i.e., T_n^{ACK}). Fig. 2 illustrates the PA and INT of the ZW policy. The ZW model proposed in [3], [6] was based on the assumption that the client would receive the ACK and generate a new status update instantaneously. However, this assumption does not represent many IoT applications accurately and holds true only in some special cases, such as point to point communication.

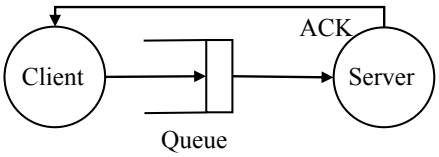

Fig. 1. Zero-Wait policy network as in [5], where the client sends the updates through the queue to the server, and the server sends an ACK to the client.

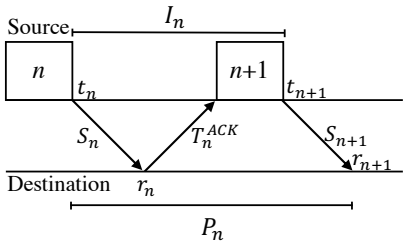

Fig. 2. An illustration of Peak Age for the Zero-Wait policy. Where t_n is the time at which update n was generated, r_n is the time update n was received, T_n^{ACK} is n ACK delay time, I represents the inter-arrival time and P_n is the Peak Age of update n.

III. When is the Zero-Wait not optimal?

To understand the limitations of ZW policy, it is important to explicitly identify the scenarios that the ZW is not optimal. Hence, it is useful to have a closed form expression for the ZW PA, as (1) represents the general case of PA. For the ZW, the waiting time is equal to zero,i.e., $W_n = 0, \forall n$. On the other hand, the inter-arrival times depend on the service time for the update and the corresponding ACK delay time (T^{ACK}). In particular, the inter-arrival time for ZW,

$$I_n^{ZW} = S_n + T_n^{ACK}, \tag{2}$$

therefore,

$$P_n^{ZW} = 2T_n + T_n^{ACK} = \frac{2}{\mu_n} + T_n^{ACK}. \tag{3}$$

Therefore, using (1) and (3) the ZW would achieve a longer PA than a general status updating policy (P) if,

$$P < P^{ZW} \; iff \; E\{w + \frac{1}{\mu} + \frac{1}{\lambda}\} < E\{\frac{2}{\mu} + T^{ACK}\} \tag{4}$$

$$P < P^{ZW} \; iff \; E\{w + \frac{1}{\lambda} - \frac{1}{\mu}\} < E\{T^{ACK}\}. \tag{5}$$

Consider an M/M/1 queue where the client updates inter-arrival time follows an exponential inter-arrival time; its waiting time is $w = \frac{\lambda}{\mu(\mu-\lambda)}$. If the waiting time is substituted in (5), the condition can be written as,

$$P^{M/M/1} < P^{ZW} \; iff \; E\{\frac{\rho}{\mu - \lambda} + \frac{1}{\lambda} - \frac{1}{\mu}\} < E\{T^{ACK}\}. \tag{6}$$

For M/D/1 queue the $w = \frac{\rho}{2\mu(1-\rho)}$ the threshold is

$$P^{M/D/1} < P^{ZW} \; iff \; E\{\frac{\rho}{2\mu(1-\rho)} + \frac{1}{\lambda} - \frac{1}{\mu}\} < E\{T^{ACK}\}. \tag{7}$$

From (6) and (7), the threshold ACK delay time τ (in which the ZW would be not optimal) for an M/M/1 and M/D/1 queue with a $\mu = 1$ is shown in Fig. 3. In other words, if the ACK delay time exceeds the τ value the ZW would fail to deliver the information as fresh as a general updating policy.

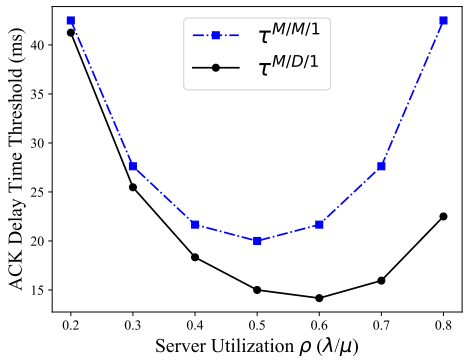

Fig. 3. The threshold ACK delay time (τ) in which updating using the PA of M/M/1 and M/D/1 queues with $\mu = 100$ is shorter than ZW.

Moreover, the ZW would fail to deliver the freshest information if the updates were sent through the internet. One the other hand, in several IoT applications the sensors transmit status updates to a server located in the cloud and which is provided by a service provider. Indeed, several IoT service providers rely on Infrastructure as a Service (IaaS) to support the sensors. As an example, Amazon provides Amazon Web Services (AWS), Google offers the Google Cloud Platform and Microsoft provides Microsoft Azure Platform. Thus, for such IoT networks, it is of paramount importance to consider the effect of ACK delay time on the PA performance. However, the proposed condition proposed in (5) is only applicable to a closed queue, and it is not applicable to the cloud scenario. Hence, to understand the limitation of the ZW in the cloud it must be evaluated experimentally.

IV. Experiment Setup

In this paper, the performance of the ZW policy in terms of PA and throughput is obtained using a Server-Client network topology for the following two scenarios: S1 and S2. In S1, both server and client were placed on the same LAN while in S2 a virtual server was used that was located on an IaaS provider and the client was located at the University of Greenwich, Medway campus.

The status updates were sent from the client to the server. Each update contained the instantaneous time-stamp representing the time of the generation of the update (t). Subsequently, the client only generated a new update and time-stamp after receiving an ACK from the server. A flowchart demonstrating the client function is presented in Fig. 4. In S1, a delay was added in the server to investigate the effect of the service time on the PA and throughput performance. The delay followed an exponential distribution. After the delay period, the server sent an ACK back to the client. In S2, as soon as the server received an update, it sent an ACK back to the client with no added delay to investigate the effect of delays arising from using a cloud-based server rather than a LAN one. Subsequently, in both scenarios, the server recorded the corresponding instantaneous time-stamp (r) for each update received and logged both the time of the generation and the receipt of the update.

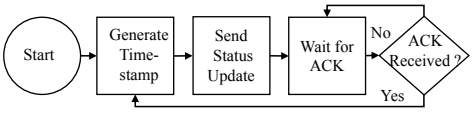

Fig. 4. Zero-Wait client flowchart. The client generates a time-stamp, sends it, then waits for an ACK and then it will repeat this procedure.

The PA and INT performance of the ZW policy was compared with the CU policy. The CU policy is a very simple policy in which the client generates a time-stamp and sends it to the server. Instantly after that, it would generate the next time-stamp. Therefore, the inter-arrival time depends on how quickly the client can generate and transmit an update (or on the client's processor clock frequency). In the experiments conducted, in S1 $\lambda \approx \mu$.

The PA and delay time of the n^{th} status update can be calculated as follows:

$$P_n = r_{n+1} - t_n \quad \text{and} \quad T_n = r_n - t_n \tag{8}$$

The PA vector is defined as $\delta = (P_1, P_2, ..., P_{N-1})$, where N is the total number of status updates sent. Similarly, the inter-arrival time vector is $i = (I_1, I_2, ..., I_{N-1})$. The experimental PA ($P$) and the value of INT, as calculated in experiments, denoted by I, is equal to the median value of the i vector, can be calculated as

$$P = \tilde{\delta} \quad \text{and} \quad I = \tilde{i} \tag{9}$$

where $\tilde{\delta}$ is the median value of the vector δ.

V. Zero-Wait Peak Age and Throughput Performance

Initially, the effect of the ACK delay on the ZW PA performance was investigated. Fig. 5, presents the PA of ZW and M/M/1 with $\mu = 100$. As proposed in (6), the PA of ZW exceed the PA of the M/M/1 queue when the ACK delay approaches the τ value. Also, it is observed that the results validate the expression derived in (3).

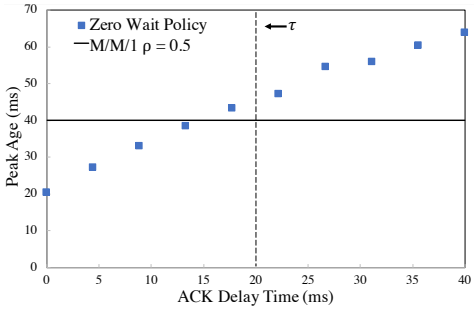

Fig. 5. ZW and M/M/1 queue, PA performance (P) versus ACK delay time.

The ZW policy shows significantly different performance for the two given scenarios. Fig. 6 and Fig. 7 show P and I respectively for 1000 updates in S1 compared with the mean service time. In S1, the Zero-Wait policy outperformed the CU policy in terms of both P and I, as shown in Fig. 6 and Fig. 7. In particular, the CU PA is 60 times longer than the ZW policy for all the service times that were tested. This considerable difference was due to the exponentially increasing waiting time of the CU policy. On the other hand, the ZW policy achieved a low waiting time. The median INT I, as a function of mean service time, was similar for both policies as shown in Fig. 7.

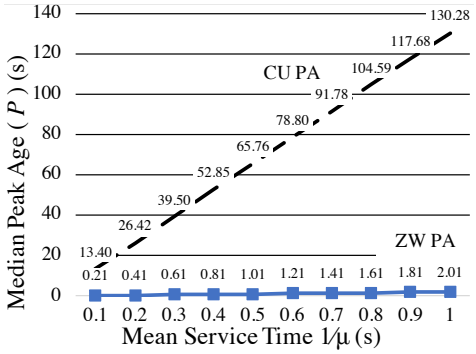

Fig. 6. ZW and CU, PA performance (P) versus service time for S1. The PA value for both policies increase with the service time, however, its value for CU is notably higher.

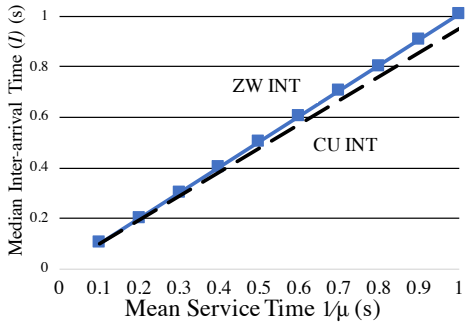

Fig. 7. ZW and CU, INT performance (I) versus service time for S1. The inter-arrival time performance of both policies is approximately equal.

The results shown in Fig. 6 and 7 are plausible and in agreement with the literature [2], [3]. In particular, the ZW policy achieved low PA and INT times. Thus, in S1, it can be assumed that the ACK service time is short enough to be neglected in comparison with the waiting and service time.

However, in S2 (when the server is in the cloud) the ACK delay time has a noticeable impact on the PA and INT times. In particular, the argument that the ZW policy is optimum in terms of PA and throughput does not hold true in S2, as shown in Fig. 8. Here both the P and I for the ZW policy are approximately twice that of the CU policy. Consequently, in this scenario, it can be argued that CU would outperform ZW in terms of both PA and throughput.

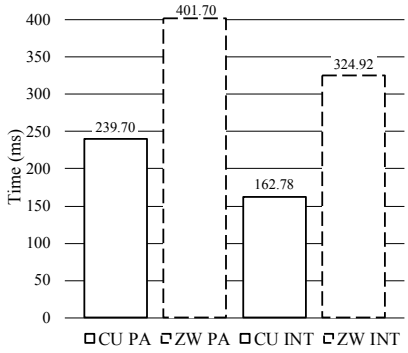

Fig. 8. ZW and CU, performance when the server is located in the cloud (S2). The CU policy outperforms the ZW policy for both the PA and the INT.

To validate the experimental results, a statistical analysis (presented in Table 1) was performed. An analysis was conducted on the delay time to make sure that the performance was not unduly affected by the fluctuations in the Internet load, which would affect the data propagation time. The analysis results are shown in Table I. It is observed that the difference in the T between the two policies is negligible (less than 0.1 ms). It can also be observed from the measures of the variance of T for both policies that the number of updates communicated was sufficient to mitigate for any Internet load fluctuations. Consequently, it is noted that the fluctuations in the T did not have a major effect on the performance of the policies.

TABLE I
STATISTICAL ANALYSIS OF THE SECOND SCENARIO RESULTS FOR DELAY TIME T, PEAK AGE δ AND INTER-ARRIVAL TIME i

Parameter	T		δ		i	
Percentile	CU	ZW	CU	ZW	CU	ZW
10% (ms)	76.57	76.64	237.70	398.30	160.80	321.50
25% (ms)	76.62	76.69	238.30	399.80	161.30	322.90
50% (Median) (ms)	76.85	76.75	239.70	401.70	162.80	324.90
75% (ms)	77.23	76.93	241.40	403.30	164.60	326.40
90% (ms)	77.49	77.23	242.50	405.10	165.40	328.30
Mean (ms)	77.02	76.86	240.20	408.60	163.20	331.70
SD	0.002	0.000	0.004	0.127	0.004	0.127

The median PA value, i.e., P, differs significantly between the ZW and CU policies. As presented in Fig. 8, the P for the CU and ZW policies are approximately 240 (ms) and 402 (ms) respectively. Consequently, it can be claimed that the CU outperforms ZW in terms of P performance. Furthermore, the median INT duration i.e., I, of the CU is approximately half that of the ZW policy. Thus, it can be seen clearly that CU outperforms ZW by a factor of two for this scenario.

TABLE II
T TEST: TWO-SAMPLE ASSUMING UNEQUAL VARIANCES

Parameter	δ		i	
Policy	CU	ZW	CU	ZW
Mean	0.24	0.41	0.16	0.33
Variance	0.000	0.016	0.000	0.016
t Stat	-41.86		-41.91	
P(T<=t) one-tail	<1%		<1%	
t Critical one-tail	1.646		1.646	

To investigate the difference in the performance of both policies, a Student t-test was performed on the PA and INT results i.e., δ and i, for both policies. As shown in Table I, the variance of the policies differed notably, hence the t-test with unequal variances was used [7]. We assumed that the hypothesis was that CU outperforms the ZW policy (in both PA and INT) is $\mathcal{H}_1$ hypothesis and the null hypothesis, $\mathcal{H}_0$ is that there is no difference between the policies. Table II shows the null hypothesis $\mathcal{H}_0$ can be rejected at the 1% level. Consequently, it can be argued that the ZW policy resulted in statistically significantly higher PA and INT times. Hence it is asserted that the ZW policy PA and INT performance in the cloud-based server scenario does not outperform the CU policy. Indeed, the CU policy outperforms it significantly and thus, the ZW policy is not optimum in this scenario.

VI. CONCLUSIONS

In this paper, it has been shown that the ZW policy is not always the optimum policy for either PA or throughput. The results presented contradict the current paradigm that ZW is always the optimum throughput policy. The next step is either to investigate policies that are able to outperform the CU in the cloud server scenario or to investigate policies that are able to outperform the ZW in the LAN scenario. The ultimate goal is to provide guidance on which policy is the most likely to be optimum for a range of known scenarios. In the case of time-critical applications, such as telehealth applications, such choices could carry significant consequences.

REFERENCES

[1] S. Kaul, R. Yates and M. Gruteser, "Real-time status: How often should one update?," 2012 Proceedings IEEE INFOCOM, Orlando, FL, 2012, pp. 2731-2735.
[2] M. Costa, M. Codreanu and A. Ephremides, "On the Age of Information in Status Update Systems With Packet Management," in IEEE Transactions on Information Theory, vol. 62, no. 4, April 2016 , pp. 1897-1910.
[3] L. Huang and E. Modiano, "Optimizing age-of-information in a multi-class queueing system," 2015 IEEE International Symposium on Information Theory (ISIT), Hong Kong, 2015, pp. 1681-1685.
[4] S. K. Kaul, R. D. Yates and M. Gruteser, "Status updates through queues," 2012 46th Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, 2012, pp. 1-6.
[5] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal and N. B. Shroff, "Update or Wait: How to Keep Your Data Fresh," in IEEE Transactions on Information Theory, vol. 63, no. 11, Nov. 2017, pp. 7492-7508.
[6] R. D. Yates, "Lazy is timely: Status updates by an energy harvesting source," 2015 IEEE International Symposium on Information Theory (ISIT), Hong Kong, 2015, pp. 3008-3012.
[7] S. S. Sawilowsky, 'The Probable Difference Between Two Means When σ 1 2 $\neq$ σ 2 2,' vol. 1, no. 2, 2002, pp. 461-472.