An Application Level Mechanism for Reliable Message Transfer in Communications between Smart Objects in Internet Platform

Ivaylo Atanasov[1], Martin Ivanov[2], Evelina Pencheva[1]
[1]Telecommunications at Technical University of Sofia, 8 Kl.
Ohridski Blvd, Sofia 1000
Bulgaria
{iia@tu-sofia.bg}

[2]Faculty of Electronics at Technical University of Varna
1 Studentska Str, Varna 9010, Bulgaria
{martinivanov@tu-varna.bg}

ABSTRACT: *In this current work, we have proposed an application level mechanism for reliable message transfer in communications between smart objects in internet platform. We have used the transaction processing mechanism for the proposed application. We further proposed the analytical relationships and the performance of transaction processing is evaluated.*

Keywords: Internet of Things, Reliable Message Transfer, Transaction Processing

Received: 1 November 2020, Revised 1 March 2021, Accepted 12 March 2021

DOI: 10.6025/jio/2021/11/2/48-55

1. Introduction

Smart objects are able to communicate with the physical world by performing limited forms of computations as well as to communicate with the outside world and other smart objects [1]. Communications between smart objects in web are conceptualized in the term "Internet of things" (IoT) [2]. The usage of Internet protocols for interconnecting smart objects is driven by the challenges of smart object networks [3].

Interoperability is an essential requirement for the smart object networks and it suits very well to the Internet protocols, which run over link layers with different characteristics. The Internet Protocol (IP) provides interoperability with existing networks, applications, and protocols, and consequently, the IP-enabled smart objects can interoperate with large number of servers, computers, and devices.

It is expected for the smart objects networks to evolve in time, allowing future applications to take full advantage of the technology. The Internet architecture allows application layer protocols to evolve independently of the underlying network protocols. Smart objects may use different wireless and wired communication technologies. Because of its layered architecture IP allows diversity of communication technologies [4].

As a pervasive technology, smart objects networks need to feature scalability and the Internet architectures have been proven regarding scalability. For smart object there is not standard transport protocol. IPenabled smart objects may communicate with other systems and devices that run IP, making use of different transport protocols.

For smart objects networks Transmission Control Protocol (TCP) provides reliable transport and thus reducing the application complexity. TCP is a complex protocol and has performance problems for high throughput data. If high throughput requirements are not important for smart object, some of the TCP mechanisms such as sliding window algorithm and congestion control are not necessary. TCP headers are large, but a compression may be used. Many TCP solutions for smart objects are designed for resource constraints, such as the delayed ACK mechanism.

User Datagram Protocol (UDP) provides a best-effort datagram delivery service i.e. the underlying IP network does its best to deliver the datagrams, but there is no guarantee that the datagrams are delivered at the destination. UDP has a very low overhead for both header size and protocol logic. This means that both the packet transmission and reception consume less energy which is in favor of the application layer data. UDP is well suited to traffic with low reliability demands, e.g. for smart objects that report data periodically within a system for home automation or eco monitoring. Since data are sent periodically, a casual packet lost is not critical as the new reading will be sent soon enough anyway.

For smart objects, any of UDP or TCP may be used and the application requirements determine the choice of transport protocol [5], [6].

In this paper, we present a simple application level mechanism for reliable smart objects communication that is based on transaction. The application protocol consists of application logic and transaction processing functionality. The transaction processing provides reliable message transfer by the use of retransmissions. We derive analytical dependences and evaluate the proposed mechanism performance.

2. Application Level Mechanism for Transaction Processing

The application logic may be described as a set of fairly independent processes with a loosely coupling between them. The processes communicate by exchange of messages. Any exchange of messages is organized in requests and responses. A request with all the responses associated with it forms a transaction. In the transaction-oriented approach, protocol requests flow from clients to a server, and the responses flow the opposite way. The reliable transmission of protocol messages within the transaction is provides by a retransmission mechanism. The transaction processing filters retransmissions in the receiving end in order to prevent the application logic from multiple message receptions.

The transactions implement a two-way handshake. *R* equests are retransmitted by the client transactions at specified intervals (*Timer* $T_{re\text{-}req}$) if a response has not been received. The duration of the whole transaction is limited (*Timer* T_{req}). The server transaction will retransmit responses if a new request arrives. Once a response has been sent by the server transaction, it will still wait for server transaction timer $T_{re\text{-}res}$ expiry to see if it receives a new retransmission of the request, which would indicate that the response has not arrived successfully. When the client transactions receives a response it terminates any additional response retransmissions related to the request are disgarded. There are two queues at the client transaction side: $Request_{queue}$ is used to store the requests, and $Timer_{queue}$ is used to store the retransmission timers.

Figure 1 shows the queuing dynamics of the client transaction. The application logic sends original requests which are forwarded to $Request_{queue}$. At the same time, the client transaction starts the timer T_{req} which limits the duration of the whole transaction. The request retransmission timer $T_{re\text{-}req}$ is also set. If no response is received and the timer $T_{re\text{-}req}$ expires, then the client transaction retransmits the request and fires the timer $T_{re\text{-}req}$ again. If a response is received, the client transaction resets both timers T_{req} and $T_{re\text{-}req}$, forwards the response to the application logic.

Figure 2 shows the queuing dynamics at the server transaction. A received request is forwarded to the application logic. A response sent by the application logic is put into the response queue ($Response_{queue}$) for transmission to the client side. If a retransmitted request is received, the respective response is put into the $Response_{queue}$ for retransmission.

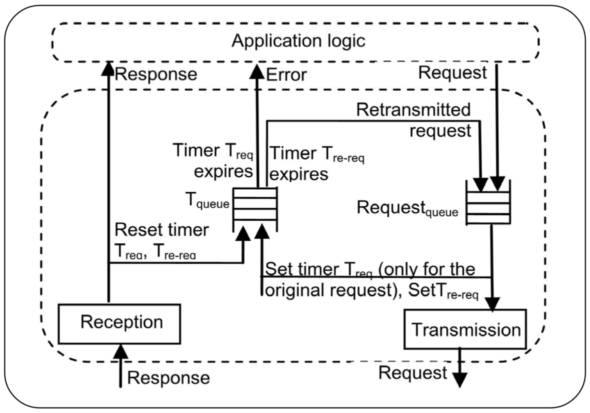
Application logic
Response
Error
Request
Retransmitted request
Timer T_{req} expires
Timer $T_{re\text{-}req}$ expires
T_{queue}
$Request_{queue}$
Reset timer T_{req}, $T_{re\text{-}req}$
Set timer T_{req} (only for the original request), Set$T_{re\text{-}req}$
Reception
Transmission
Response
Request

Figure 1. Queuing dynamics at the client transaction

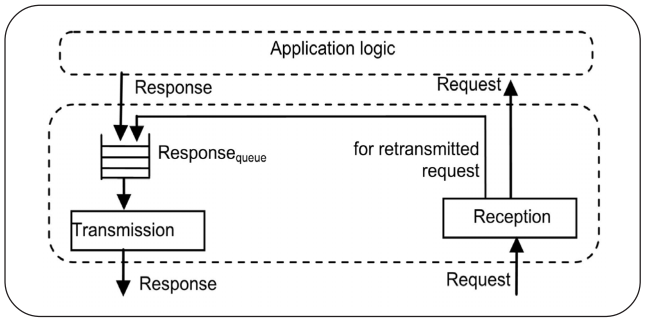

Figure 2. Queuing dynamics at the server transaction

3. Probabilistic Transaction Evaluation

Successful transaction is the transaction for which both the request and the response have arrived successfully.

Figure 3 shows the trellis diagram of the transaction states where P_{loss} is the probability of message loss (request or response). In Figure 3, S_0 is the initial state and the state S_3 corresponds to the case where two consecutive request transmissions failed, and the third one leads to a successful response.

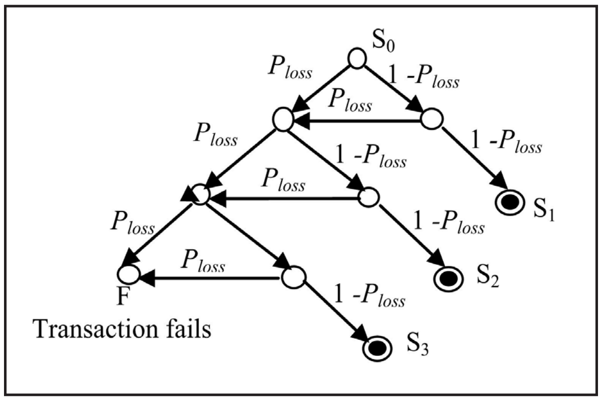

Figure 3. Trellis diagram of transaction states limited to 3 requests transmissions

Let us denote by P^k_{trans} the probability of successful transaction after k request sending (i.e. to reach state S_k) is:

$$P^k_{trans} = (1 - P_{loss})^2 P^{k-1}_{loss} (2 - P_{loss})^{k-1} \quad (1)$$

The probability $P_T(P_{loss}, N)$ of successful transaction for any of the N possible request transmissions is as follows:

$$P_T(P_{loss}, N) = \sum_{k=1}^{N} P^k_{trans} = (1 - P_{loss})^2 . \sum_{k=1}^{N} P^{k-1}_{loss}(2 - P_{loss})^{k-1} \qquad (2)$$

Figure 4 shows the dependence of P^k_{trans} as a function of P_{loss}.

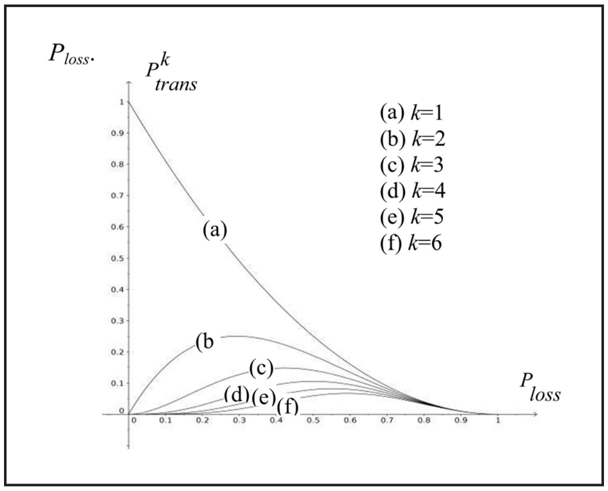

Figure 4. Probability of successful transactions with just k request transmissions as a function of probability of message loss

Figure 5 shows the dependence of the probability of successful client transaction as a function of the probability of the loss of messages. In the figure, case (a), (b) and (c) correspond to the values of N = 5, 4, 3 in (2), case (d) and (e) show the difference in the probability of a successful transaction, respectively, for N = 5 and 43, and for N = 4 and 3.

Figure 5 highlights the values of probability of successful transaction for message loss probability of 0.3, respectively for 5 requests transmissions $P_T(P_{loss}, N) = 0.98223$, for 4 requests transmissions $P_T(P_{loss}, N) = 0.96522$ and for 3 requests transmissions $P_T(P_{loss}, N) = 0.93191$.

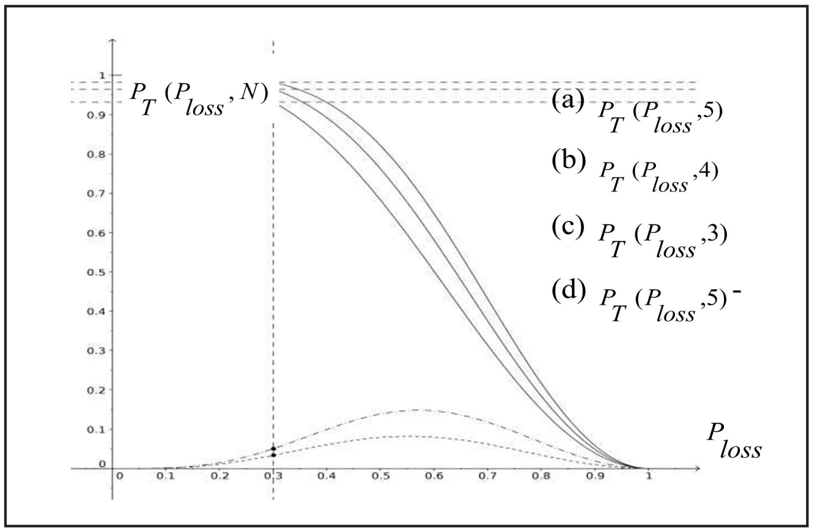

Figure 5. Dependence of the overall probability of a successful transaction as a function of probability of message loss

Accordingly, the difference between the probabilities for 5 and 4 requests transmissions is 0.03331, and the difference between the probabilities for 4 and 3 requests transmissions is 0.05032.

One of the parameters to assess the effectiveness of the proposed mechanism for transaction processing, is its latency.

Let *d* denotes the delay in message transmission (request or response). If the first request transmission is unsuccessful, the request is retransmitted after expiry of timer T_1. If the *k* -th request transmission is unsuccessful, the request is retransmitted after expiry of timer T_k.

The delay introduced to reach a state of successful transactions with *k* request transmissions is:

$$\delta_\kappa = 2.d + T_k \tag{3}$$

If the retransmission timer has a constant value of *T*, then

$$T_k = (\kappa - 1).T \tag{4}$$

and the delay introduced to reach a state of successful transactions with *k* number of request transmissions is:

$$\delta_\kappa = 2.d + (\kappa - 1)T \tag{5}$$

The probability estimation of the average latency of successful transaction in exactly *k* requests transmissions is expressed as follows:

$$\begin{gathered}\Delta_\kappa(P_{loss}, d, T) = \delta_k.P_{trans}{}^k = \\ (2.d + (k-1).T)(1 - P_{loss}{}^2).P_{loss}{}^{k-1}.(2 - P_{loss})^{k-1}\end{gathered}. \tag{6}$$

Hence, the probability estimation of the average latency of successful transaction is:

$$\begin{gathered}D(P_{loss}, N, d, T) = \sum_{k=1}^{N} \Delta_k(P_{loss}, d, T) = \\ \sum_{k=1}^{N} (2.d + (k-1).T)(1 - P_{loss}{}^2).P_{loss}{}^{k-1}.(2 - P_{loss})^{k-1}\end{gathered} \tag{7}$$

Let fix the delay in the network $d = 50$*ms*, and the value of the retransmission timer is $T = 800$ ms.

The family of curves for $k = 1..\ 6$, shown in Figure 6, present the probability estimation of the average latency of successful transaction in exactly *k* requests transmissions.

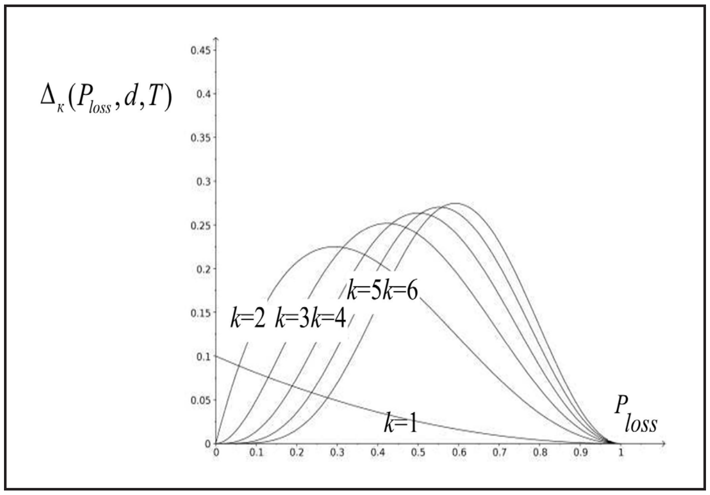

Figure 6. Probability estimation of the average latency of successful transaction for exactly *k* request *s* transmissions ($d = 50$ *ms*, $T = 800$ *ms*)

The family of curves for $N = 4$ ..7, shown in Figure 7, present the probability estimation of the average latency of successful transaction where possible request transmissions in a transaction are N and the value of the request retransmission timer is constant.

The family of curves, shown in Figure 8, present the probability estimation of the average latency of successful transaction, where $N = 4$ and the value of the retransmission timer T varies from 300 ms to 800 ms.

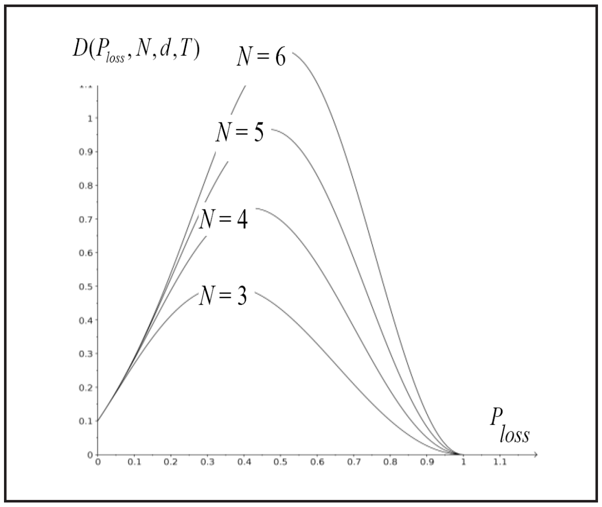

Figure 7. Probability estimation of the average latency of successful transaction with N possible request stransmissions ($d = 50$ ms, $T = 800$ ms)

If the request retransmissions take place due to congestion in the network, then it makes sense for the request retransmission timer to increase at each retransmission.

In case of network congestion, the increase of the retransmission timer value for each subsequent retransmission would increase the probability of a successful transaction.

Let us denote by $I(s)$ a function that has value 1 if s is true, or the value 0 otherwise. If the value of the request retransmission timer T doubles with each successive request retransmission, then the probability estimation of the latency of successful transaction with N possible transmission of the requests is:

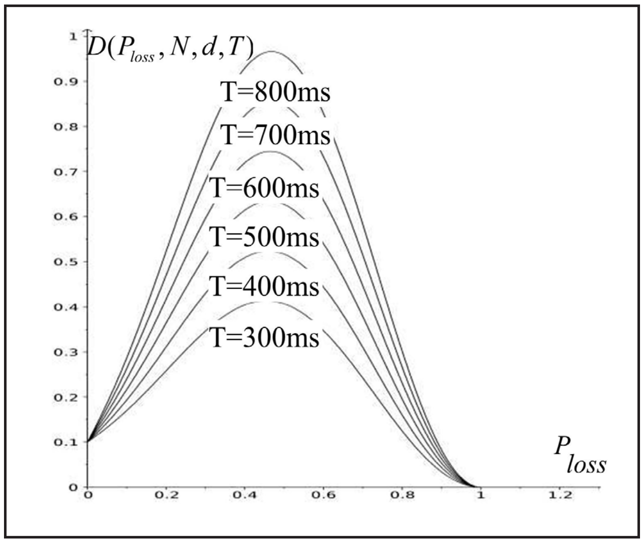

Figure 8. Probability estimation of the average latency of successful transaction (d = 50ms, N = 4)

$$D(P_{loss}, N, d, T) = \sum_{k=1}^{N} \Delta_k (P_{loss}, d, T) =$$
$$\sum_{k=1}^{N} (2.d + I(k > 1).T.2^{k-1})(1 - P_{loss}{}^{2}).P_{loss}{}^{k-1}.(2 - P_{loss})^{k-1} \qquad (8)$$

The value of the request retransmission timer can be limited. The smaller upper threshold of the retransmission timer leads to transaction behavior that resembles the constant value of the retransmission timer.

The bigger upper threshold of the retransmission timer leads to transaction behavior that is similar to the behavior of non-limited value of the retransmission timer. Let us denote by *L* the maximum value of the request retransmission timer. Then $\Phi(k, T, L)$ represents a function defined in the following manner:

$$\Phi(k,T,L) = \begin{cases} 0 \text{ , where } k \le 1 \\ T.2^k \text{ , where k is such that } T.2^k \le L \\ L\text{, otherwise.} \end{cases} \qquad (9)$$

Then the probability estimation of successful transaction latency with a doubling of the retransmission timer for any subsequent transmission up to a maximum value *L* is given by the following:

$$D(P_{loss}, N, d, T, L) = \sum_{k=1}^{N} (2.d + \Phi(k,T,L))(1 - P_{loss}{}^{2}).P_{loss}{}^{k-1}.(2 - P_{loss})^{k-1} \qquad (10)$$

Figure 9 and Figure 10 show a comparison of the probability estimation of the successful transaction latency for the case of a constant value of the retransmission timer for the query (a), the case of a exponential increase of the retransmission timer value without limitation on the maximum value (b), and the case (c) with maximum value of upper requests retransmission timer *L* respectively of 1500 ms and 3000 ms. In the figures, the fixed number of request retransmissions *N* is 6.

4. Conclusion

In this paper an application level mechanism for reliable message transfer in internet-based communications between smart objects is proposed. The mechanism is based on transactions and its performance is evaluated. The results show that when determining the threshold for the maximum value of the request retransmission timer, a compromise between a shorter duration of successful transaction with a focus on request retransmissions in fixed intervals and greater probability of successful transaction with exponential increase of time for request retransmission might be sought.

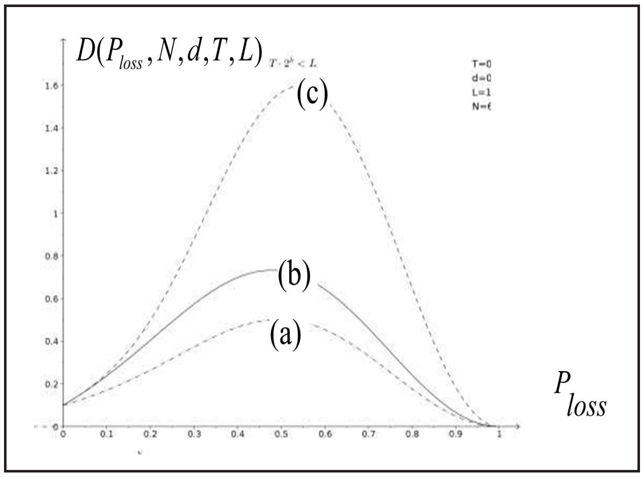

Figure 9. Probability estimation of the average latency of successful transaction ((a) T = const; (b) $T = T.2^k$; (c) T = min $(T.2^k, L)$ L = 1500 *ms*

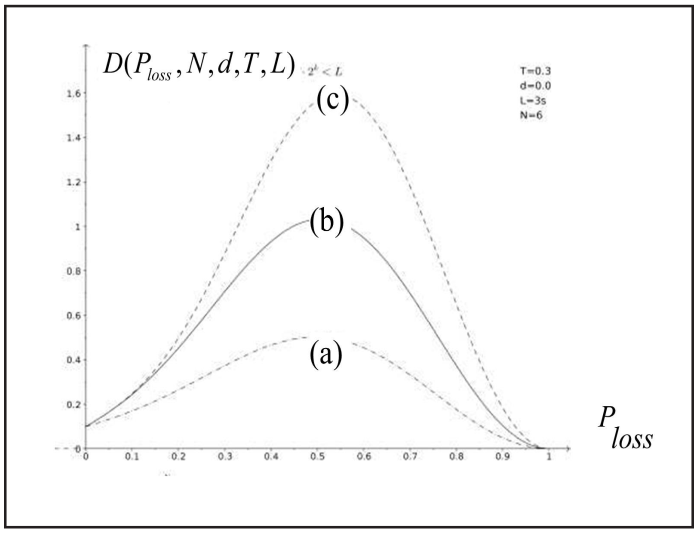

Figure 10. Probability estimation of the average latency of successful transaction ((a) T = const; (b) $T = T.2^k$; (c) $T = min(T.2^k.L)$, L = 3000 *ms*

Acknowledgement

The work is under the support of the research project №141ПР004-7 funded by RDS, TU-Sofia.

References

[1] Vasseur, J., Dunkels, A. (2010). *Interconnecting Smart Objects with IP. The Next Internet*. Morgan Kaufmann.

[2] Atrozi, L., Piera, A., Morabito, G. (2010). The Internet of Things: A survey, Computer Networks, Volume 54, 2787-2805.

[3] Chaouchi, H. (2010). Introduction to the Internet of Things. In H. Chaouchi *the Internet of things. Connecting Objects to the Web*. Wiley.

[4] Ivanov, M., Dimova, R. (2014). PMU traffic evaluation in wide area monitoring and control systems, *Computer Sciences and Communications*, BFU, 3 (1) 3-11.

[5] Elgazzar, K., Aboelfotoh, M., Martin, P., Hassanein, H. (2012). Ubiquitous Health Monitoring Using Mobile Web Services, *Procedia Computer Science*, 10, 332-339.

[6] Tarouco, L., et. al. (2012). Internet of Things in healthcare: Interoperability and security issues, IEEE ICC'2012, Conference Proceedings, 6121-6125.