Analytical characterization of failure recovery in REAP

A. de la Oliva [a,*], I. Soto [a], A. García-Martínez [a], M. Bagnulo [a], A. Azcorra [a,b]

[a] *Departamento de Ingeniería Telemática, Universidad Carlos III de Madrid, Spain*
[b] *IMDEA Networks, Avda. Mar Mediterraneo, 22 Leganes, Madrid 28918, Spain*

ABSTRACT

This paper characterizes analytically the performance of REAchability Protocol (REAP), a network layer end-to-end recovery protocol for IPv6. REAP was developed by the IETF SHIM6 Working Group as part of its multihoming solution. The behavior of REAP is governed by a small number of parameters: three timers, a simple characterization of the application traffic, and the communication delay. The key figure of merit of REAP performance is the time to recover from a path failure as seen by the upper layers, figure that cannot be trivially obtained, despite the apparent simplicity of this reachability protocol. In this paper we provide upper bounds for the recovery time of REAP for different deployment scenarios, applying these analytical results to two interesting case studies, TCP and VoIP traffic.

The SHIM6 (Site Multihoming by IPv6 Intermediation) Working Group[1] of the IETF has developed a framework that enables scalable fault tolerance protection for on-going communications in IPv6 multihomed environments. Considering that the large address space of IPv6 allows end hosts to configure as many addresses as available providers, this framework aims to enable the use of these different addresses for a single communication which enables the use of different paths. The address agility function is performed by a shim sublayer, named SHIM6, defined inside the IPv6 layer. This SHIM6 sublayer manages the mapping between the addresses being exposed to the upper layers, which remain constant during the communication lifetime, and the addresses included in the packets sent through the wire, that could vary and enforce the use of different paths. The SHIM6 protocol [1] creates and manages these mappings between the SHIM6 sublayers of the two nodes involved in the communication.

A fault tolerance solution requires a mechanism to detect failures across the communicating path, and a mechanism to discover a valid path after a failure. In particular, the mechanism should allow transport layer survivality, to be fully transparent for transport-layer sessions [2]. The SHIM6 Working Group defines such a component, named REAP (REAchability Protocol, [3]), which detects failures in any of the two unidirectional paths in use for a communication, and explores different unidirectional paths to find a valid one after an outage. Note that a bidirectional path is modeled by REAP as two unidirectional paths.

The REAP instance of an endpoint detects a failure by monitoring the packets received for a given communication. When a communication involves a bidirectional exchange of data at a sufficient rate, the availability of the path is determined without exchanging REAP-specific packets. If one of the endpoints is not sending data regularly or the if the rate at which data is being sent is too low, its REAP entity generates Keepalive messages that prevent the expiration at the other end of the timer used to detect failures. When no party sends upper layer data for some time REAP stops generating Keepalive messages and failure monitoring is suspended.

When a failure is detected, REAP triggers the path exploration function. The currently used unidirectional paths are initially tested by sending REAP Probe messages. If this validation fails, Probe messages with different combinations of source/destination addresses are sent until a new pair of working addresses is found. Note that SHIM6 and REAP support the use of paths defined by different source and destination address pairs in each direction.

Ideally, a failure detection mechanism should require as low resources and bandwidth as possible. The amount of state required for REAP operation is just three timers per communication and per endpoint. Additionally, it is quite efficient in terms of the number of protocol-specific messages exchanged since Keepalive messages are only sent for unidirectional or low-rate communications.

REAP is a good solution to provide a failure detection and path exploration mechanism to other protocols requiring such

* Corresponding author. Tel.: +34 916248803.
E-mail addresses: aoliva@it.uc3m.es (A. de la Oliva), isoto@it.uc3m.es (I. Soto), alberto@it.uc3m.es (A. García-Martínez), marcelo@it.uc3m.es (M. Bagnulo), azcorra@it.uc3m.es (A. Azcorra).

[1] http://www.ietf.org/html.charters/SHIM6-charter.html

functionality, because it has minimal requirements and it is independent from the SHIM6 protocol. Examples of protocols that could benefit from this functionality are Host Identity Protocol (HIP) [4] [5], Mobile IPv6 with registration of multiple CoAs (Care-of Address) [6], Mobike (IKEv2 Mobility and Multihoming Protocol) [7] or combined SHIM6/Mobile IPv6 operation [8].

Although two simulation and experimental studies have been previously published, focusing either on the path exploration process [9] or on the impact of the transport protocol on the recovery time [10], no analytical characterization of the time required to recover from a failure has been provided so far. Note that this value is a key figure of merit for determining the impact perceived by upper layers. Too large recovery times can result in the communication being discarded by the upper layers. But even if the communication continues, the quality can be degraded if the recovery process takes longer than the time required by the application. Proper characterization of REAP performance would enable the configuration on a per communication basis of the REAP timers in order to comply with specific upper layer constraints.

However, despite the functional simplicity of the REAP mechanism, the characterization of the recovery time is far from trivial. It could be initially thought that the time required to detect a failure by REAP depends only on the value of the three defined timers, the Send Timer, the Keepalive Timer and the Retransmission Timer, but this would obviate the relation of the time at which these timers are started and the time at which the failure occurs. This relation largely depends on the specificities of the communication. Without a proper computation of this time it is not possible to provide an upper bound which can be used by the applications to cope with the failures in the communication.

In this paper we characterize analytically the upper bound of the time required by REAP to detect a failure and recover from it in different scenarios. These results are applied to different traffic patterns and to the specific case of TCP as the transport protocol.

The remainder of the paper is organized as follows: Section 2 provides an insight on the REAP protocol. In Section 3 we present the reference model used for the analysis, and we detail the definition of the Recovery Time. In Section 4 we start a top/down characterization by analyzing the contributions to the Recovery Time of the failure detection process and the exploration process. For this analysis, different types of communication (bidirectional and unidirectional) and failure (Two-Way and One-Way) are considered. Section 5 is devoted to characterize the τ parameter, that depends on the time elapsed between the transmit time of the first lost packet in any node and the starting time of the Send Timer. τ is the main parameter to be considered when estimating the duration of the detection process. To obtain the least upper bound of τ, the maximum value among several cases has to be considered. The methodology followed to obtain the upper bound of τ is verified by simulation results. Then, in Section 6, we provide more compact expressions for the combination of the results obtained in Sections 4 and 5. These expressions eliminate the dependency on the failure type or location, and are the results to be used for configuring REAP to comply with the specific requirements of an application. An applicability example of the results is presented in Section 7. To give further details of the applicability of the analytical results, we consider the variable rate traffic case and applications that use TCP as transport-layer in Section 8. Finally, in Section 9 we present the conclusions and future work.

2. Failure detection and path exploration in REAP

In this section we describe in detail the two components of REAP [3], the failure detection and the path exploration mechanisms. The failure detection mechanism of REAP is used to monitor the status of the pair of unidirectional paths active in a communication. Note that although SHIM6 is able to manage alternative paths for a communication, REAP only tests the pair of paths in use at a given time. To validate the current two unidirectional paths of a communication, REAP relies on two timers in each node, the Keepalive Timer and the Send Timer, and, when required, on the exchange of a message, named the Keepalive message. The Keepalive Timer is started each time a node receives a data packet from its peer, and stopped and reset each time the node sends a packet to the peer. When the Keepalive Timer expires, a Keepalive message is sent to the peer. Note that the reception of Keepalive messages does not modify the value of the Keepalive Timer at the receiving node. The Send Timer is started each time the node sends a packet, and stopped and reset each time the node receives a packet from the peer, either a data packet, or a Keepalive message. If the Send Timer expires, i.e. no packet has been received during this period, a failure is assumed and the node starts the path exploration process.

The Send Timer expiration indicates that no return traffic was received for some time by a node that was sending data. On the other hand, the Keepalive Timer is used to assure that return traffic, in this case Keepalive messages, is generated in nodes that are receiving data but have no data to send. Note that the values of the Keepalive Timer and the Send Timer should allow at least one Keepalive message to arrive to the destination to avoid false failures. The current specification suggests a default value of 15 s for the Send Timer, while no value is proposed for the Keepalive Timer. Note that neither experimental data nor analytical studies have been considered to propose the values for any of the timers that determine the performance of REAP.

REAP does not include any mechanism for detecting congestion. Severe congestion in the network is considered by REAP as a failure. If packets stop reaching the node for a Send Timer period, due to congestion, REAP assumes a failure has occurred (a false positive) and starts the exploration mechanism. We argue that this is an appropriate behavior when network congestion is such that the time during which the path is unavailable exceeds the threshold set by the application using it.

Once a node detects a failure, it starts the path exploration mechanism by changing its state from Operational to Exploring. First, a Probe Exploring message (a Probe message with the Exploring flag set) is sent to test the current address pair. This allows resuming the communication through the initial path after short unavailability periods, due for example to light network congestion or local route reconfiguration. In this case REAP completes the required handshake through the current path and returns to the Operational state without disrupting the communication. However, if no response is obtained during a Retransmission Timer period, alternative outgoing paths, defined by different combinations of source and destination addresses, are tested by sending Probe Exploring messages and waiting for a response during a Retransmission timer period. In the current specification, only one Probe is sent at a time. After sending four Probe Exploring messages, an exponential backoff algorithm increases the Retransmission Timer. When a Probe Exploring is received, this means that a valid unidirectional path has been discovered for the incoming path. The node that has received the Probe Exploring message then changes its state to Inbound_OK and uses Probe Inbound_OK messages to continue exploring outgoing valid paths. This type of Probe messages includes an indication of the valid incoming path. If the other node receives a Probe Inbound_OK, it can assume as valid the incoming path through which the packet was received, and it can obtain from the payload of the Probe Inbound_OK the valid outgoing path to be used. Then, the node changes its state to Operational, and sends a Probe Operational message in which it informs its peer about the validity of the path through which it received the Probe

Inbound_OK message. A node that receives a Probe Operational message changes its state to Operational. It is worth to highlight that data is still being sent when the node is in the Exploring or the Inbound_OK states using the source and destination addresses in use when the node was in the Operational state. When the Operational state is reached again, the addresses in use are changed to the ones resulting from the exploration process. Note that these rules may lead to different state and message sending schedules: for example, one node can detect a failure and send a Probe Exploring that arrives to its peer before the peer detects a failure; or both nodes can detect a failure before receiving a Probe from the other endpoint. Fig. 1 presents the state machine diagram that formalizes the behavior described above, including some transitions that occur only when a limited number of packets (either data, Keepalive or Probes) are lost, which could occur due to temporary path unavailability. Although, as already mentioned, in the current specification Probe messages for exploring alternative paths are sent sequentially during the exploration phase, in [9] a concurrent exploration mechanism of alternative paths is defined. This concurrent exploration is done after probing the current path.

It could be thought that we could reduce the time to recover from failure by exploring the alternative paths also concurrently with the probing of the current one when the REAP state changes from Operational to Exploring. While this could be true, enforcing the preference to select other locators only if the Probe of the current path has failed at the same time at which other paths are been concurrently probed may complicate slightly the state machine of the REAP entities. For this kind of operation, REAP should store the addresses of the first received Probe (if different from the current incoming path), and wait for a Probe for the current incoming path until the Retransmission Timer expires. If the Probe from the current incoming path arrives in time, the current path is preserved, and otherwise the probing process continues with the information of the first received Probe. In the rest of the paper we do not consider this mode of operation (not mentioned also in the REAP specification [3]), although the analysis could be easily adapted to it.

3. Model for performance evaluation in REAP

In this section we present the reference model to be used in the performance analysis for REAP. We first discuss the parameters involved in the failure detection and recovery procedures. Then, we define the figure of merit through which we evaluate the performance of REAP, the Recovery Time.

3.1. Reference Model for REAP

Consider two nodes that are communicating. Packets traveling from A to B, and from B to A, experience a fixed end-to-end delay of γ_{AB} and γ_{BA} respectively which may change when the addresses in use change. Note that γ_{AB} and γ_{BA} can be quite different for many reasons. To name a few, they can be different because SHIM6 allows each direction to be determined by unrelated source and destination addresses, resulting in completely different unidirectional paths, or because even if the same routers were traversed in both directions, queuing delay can be different as a result of different traffics being served for each segment on each direction. The Round Trip Time at a given time is computed as $R_{TT} = \gamma_{AB} + \gamma_{BA}$. Upper layers periodically deliver data packets to the IPv6 layer. The size of each packet is irrelevant for the analysis. Communication can be bidirectional, with node A sending data with a fixed inter-packet interval of Δ_A, and node B sending data with Δ_B interval. If the communication is unidirectional, we assume that it is node A the one that sends traffic with inter-packet interval Δ_A. The case in which no packets are sent by neither of the peers is not considered, since REAP does not perform failure detection in this case. For the rest of the definition of the model we assume bidirectional traffic without loss of generality.

A failure occurs at a given time T_{fail}. The failure could affect both directions if it is caused by a failure in an element (router or link) shared by both paths, or it can just affect to one direction (either affecting the path from A to B, or from B to A). To precisely characterize the location of the failure so that we can determine which packets were able to go through before the failure and which pack-

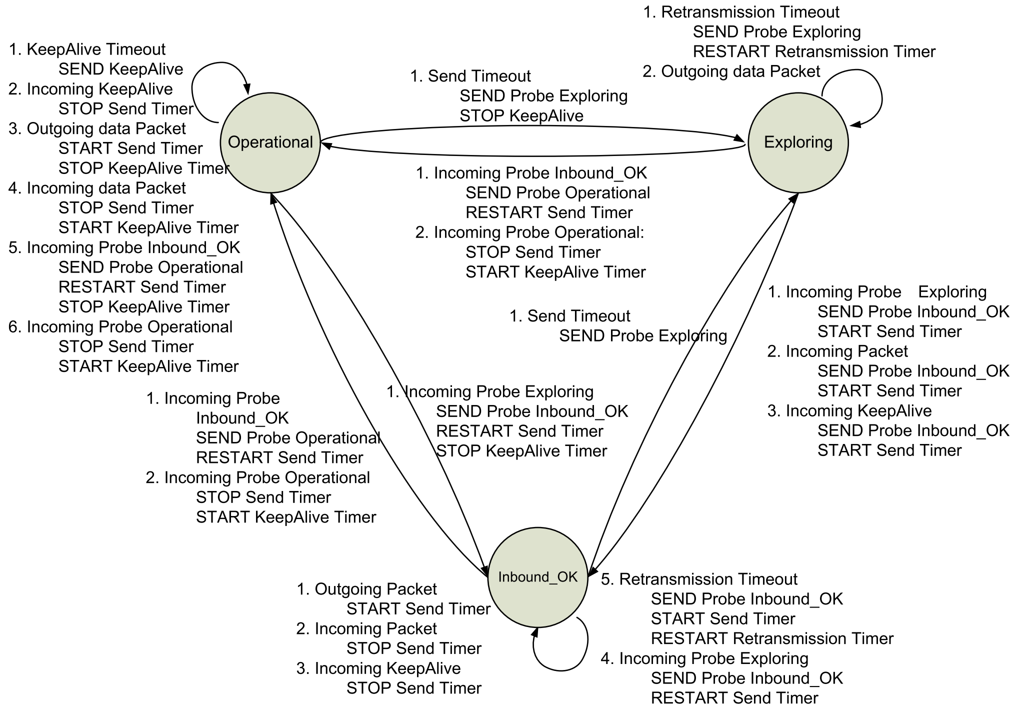


Fig. 1. State machine of REAP.

ets were affected by it, we use α and α' to define the fraction of the end-to-end delay that a packet should experience until it reaches the point at which the failure has occurred. The valid range for α and α' is $(0, 1)$. Extreme cases for α (α') of 0 or 1 would mean that the packet is dropped just when issued to be sent by the application (0) or just when it is to be delivered to the remote application (1). There are a number of reasons to have values close to these figures: a failure in the outgoing interface or in the remote incoming interface, address configuration problems in the nodes, operating system misconfiguration, etc. Note that in general α does not need to be equal to $1 - \alpha'$. Considering a bidirectional failure, a packet sent by A would spend $\alpha * \gamma_{AB}$ until it reaches the failure point, and a packet sent from B would spend $\alpha' * \gamma_{BA}$ until it arrives to the outage (see Fig. 2). Then, we can also state that packets sent from peer A at a time $T = T_{fail} - \alpha\gamma_{AB}$ or later are dropped, as well as packets sent from B later than $T = T_{fail} - \alpha'\gamma_{BA}$. Note that α and α' are unrelated because of the different properties of the communication paths in both directions.

The time at which the first lost packet from A is sent is named T_{lostA}. The time at which the first lost packet from B is sent is denoted as T_{lostB}. The possible range of values for T_{lostA} and T_{lostB} are presented in Eqs. (1) and (2).

$$T_{fail} - \alpha\gamma_{AB} \leqslant T_{lostA} < T_{fail} - \alpha\gamma_{AB} + \Delta_A \tag{1}$$

$$T_{fail} - \alpha'\gamma_{BA} \leqslant T_{lostB} < T_{fail} - \alpha'\gamma_{BA} + \Delta_B \tag{2}$$

Finally, we assume that the Send and Keepalive Timers of A and B are equal for both nodes, with values T_{Send} and T_{KA} respectively.

3.2. Recovery time

The analysis presented in this paper aims to characterize the impact of REAP on upper layers when an outage occurs. For this purpose, we define the *Recovery Time* as the difference between the time at which the first data packet lost (at any node) is sent, and the time at which every peer willing to send traffic is ready to send packets again (i.e. the peer or peers with traffic to send have returned to the Operational state). In particular, for unidirectional traffic only the peer sending traffic has to return to the Operational state to restore the original communication.

Fig. 3 shows the Recovery Time for a bidirectional data exchange with a failure affecting both directions. For the sake of clarity, data packet exchanges during the exploration process have been omitted. In the situation depicted, the Send Timer at B expires before the Send Timer at A, so it is B the node that starts probing the current path from B to A. Before any Probe Exploring arrives to A, the Send Timer at A expires, so that it also starts testing the current path from A to B. A Retransmission Timer time after the first Probe was sent, B realizes that the current path is not valid, and starts probing alternative addresses. The first alternative path tested succeeds, so A receives the Probe Exploring message, and issues a Probe Inbound_OK that includes information confirming the validity of the new path from B to A. Upon the successful reception of this message at B, B changes its state to Operational and data packets can be sent again. Finally, a Probe Operational from B to A is used to inform A that the path it had selected is valid. In the example considered, the Recovery Time is the time since the first packet sent by A was lost, until both peers return to Operational state.

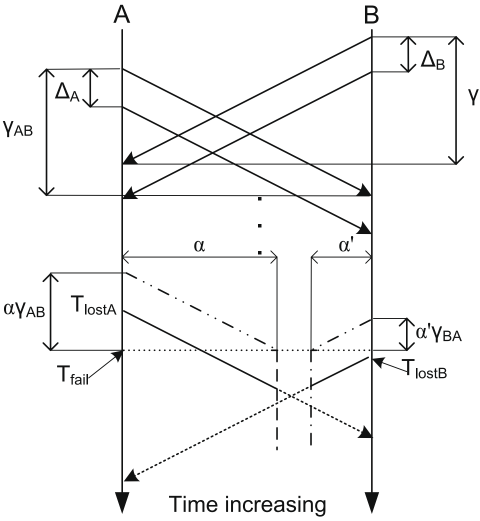


Fig. 2. Reference model for the analysis of REAP.

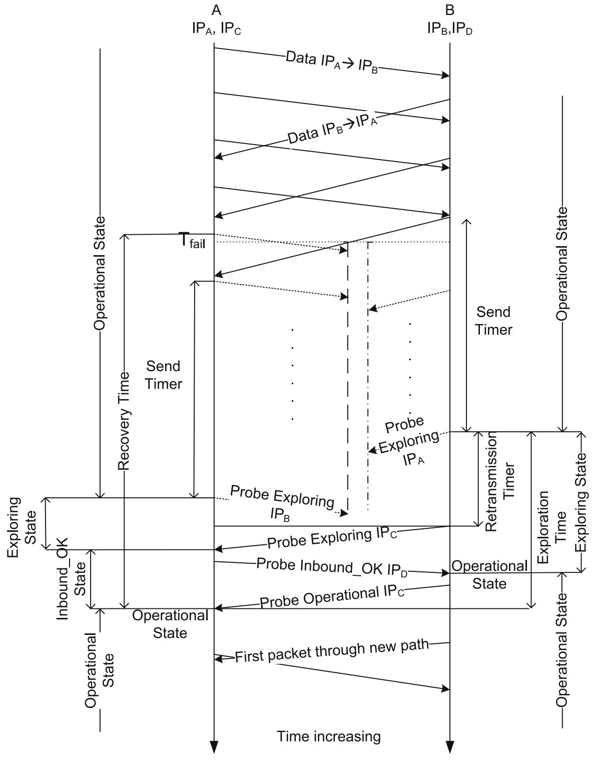


Fig. 3. Recovery time components.

As discussed before, the operation of some upper layers may be negatively affected by outages lasting for more than a given threshold, threshold that may vary for different transport and application layer combinations. Consequently, we are particularly interested in being able to estimate the upper bounds for the Recovery Time in any particular scenario determined by the type of communication (bidirectional, unidirectional), the frequency at which data packets are sent in both communicating peers, the Send Timer and Keepalive Timer values, and the end-to-end delay at both directions of the communication. Provided that the parameters characterizing the communication were known, it could be determined if a given configuration fulfills the requirements of the upper layers regardless the particular execution details such as the exact time at which packets are sent at each side or the failure details, i.e. regardless the failure affecting one or both directions or the physical location at which the failure occurs.

4. Characterization of the recovery time

In order to characterize the behavior of the Recovery Time ($T_{recovery}$ hereafter), we have to consider all possible communication scenarios that may result from the type of communication.

First, we must separate the analysis of bidirectional and unidirectional traffic. For bidirectional traffic we assume that the packet rate is high enough, precluding a Keepalive messages exchange. Additionally, we have to consider that on each peer runs a different Send Timer, resulting in different event sequences. These event sequences cannot be modeled by considering two independent unidirectional flows. The scope of the failure, either in both directions, or just in one direction, also determines different behaviors for the REAP entities involved.

Consequently, the cases that must be analyzed are:

- Bidirectional traffic, Two-Way failure.
- Bidirectional traffic, One-Way failure.
- Unidirectional traffic, Two-Way failure.
- Unidirectional traffic, One-Way data path failure.
- Unidirectional traffic, One-Way return path failure, (i.e. the path through which Keepalive messages are exchanged). Although this scenario is possible, it is not relevant for our work since this kind of failure does not affect the data exchange.

Regarding the path exploration phase, we assume that after checking the current path to confirm the failure, the possible alternative paths are explored concurrently (as recommended in [9]), and that at least one path is working (Recovery Time is meaningless if no working path is available).

Concurrent path exploration is the fastest way of providing a valid path through which the application can resume its operation, which is required in the case of applications with tight constraints in the required recovery time. However, our analysis is also valid for sequential path exploration if the first alternative path explored is working, and it can be easily extended to sequential path exploration recovering in the *n*th path.

The REAP state machine (Fig. 1) includes some logic to recover from REAP-specific packets lost due to short-lived congestion or other transient effects. In our analysis, we simplify this state machine by taking into account only state transitions that conduct to a recovery using a new path, i.e., ignoring state transitions in which REAP decide that the original path is working and therefore finally does not influence the application chosen path. So we can assume the following:

- After a failure, a node in Exploring state never receives data packets through the original path. Data packets could only be received if the node had moved to the Exploring state due to a transient loss of packets.
- Keepalives are only sent in the Operational state. If packets are not lost unless a failure occurs, a node is in Exploring state only if the original incoming path failed. Then, Keepalives should not be received through the original incoming path (because it is not available), nor through other path (because a Probe Inbound_OK should have been received before the Keepalive).
- Similarly, Keepalives should not be received in Inbound_OK state because Probe Inbound_OK or Probe Operational messages should have been received before.
- A node can only send Probe Operational messages if it is in the Inbound_OK state and has received a Probe Inbound_OK message. A peer can only send Inbound_OK messages from the Inbound_OK state. Therefore, a node in the Exploring state cannot receive a Probe Operational message.
- As at least one of the explored paths after a failure is valid and the Send Timer is large enough, this timer will not expire in the Inbound_OK state.

Taking these assumptions into consideration, Fig. 4 presents the simplified state machine of REAP. We use this state machine to derive in the following subsections the feasible transitions for each of the scenarios presented above. It is worth to note that when a failure occurs, the first event is always the expiration of the Send Timer in any of the nodes, triggering the generation of a Probe Exploring message. Then, only two possibilities are available for the transitions on the peer node: the peer remains in Operational state until it receives the Probe Exploring message, or, only in the case of bidirectional communication with Two-Way failure, the Send Timer could expire before receiving the Probe Exploring message. For analyzing the rest of the possible state transitions, we consider the type of traffic and the type of failure. For the sake of clarity, in the following explanations we are only taking into account the messages exchanged with the peer through the alternate fastest path (concurrent exploration), which is equivalent to assume sequential exploration in which the first alternate path explored is valid.

4.1. Bidirectional traffic, Two-Way failure

On this scenario, the peers are exchanging bidirectional traffic when both unidirectional paths in use are affected by an outage.

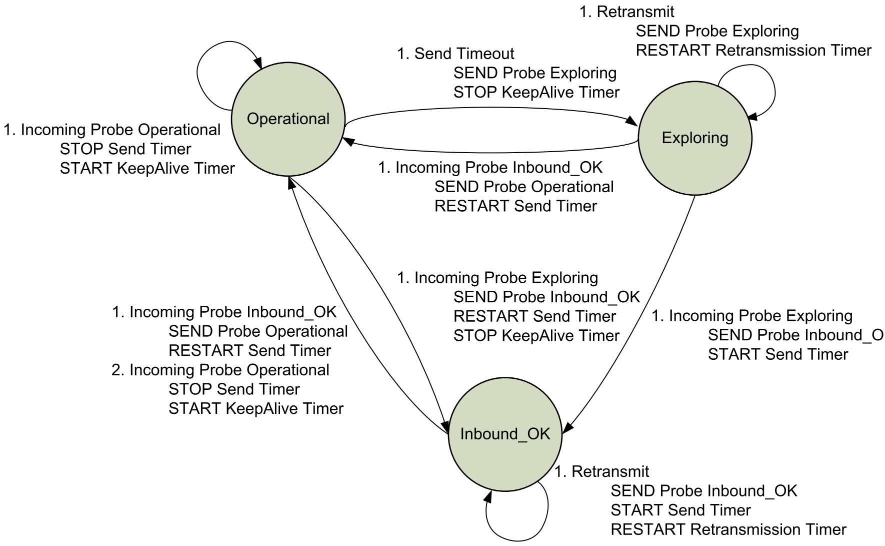


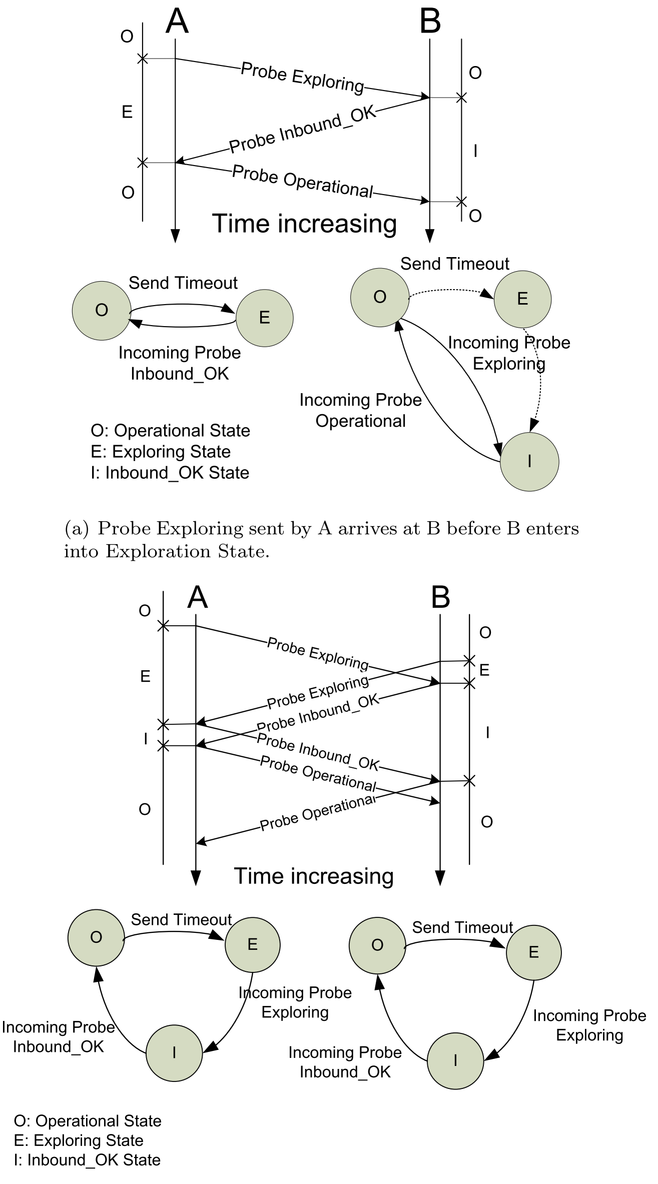
Fig. 4. REAP simplified state machine.

The restriction that a node can only reach the Inbound_OK state when the peer is in the Exploring state defines three possible state sequences in the peers until they return to the Operational state (Fig. 5):

1. In the first case, presented in Fig. 5a (continuous lines), the node discovering the failure after the expiration of the Retransmission Timer, A, sends a second Probe Exploring message that arrives to its peer B when it is in Operational state. Note that this message is able to reach node B since it is exchanged through an alternative path. The reception of the Probe Exploring results in a change to Inbound_OK in B. In this transition, B sends a Probe Inbound_OK message, which after reception triggers a transition in A from the Exploring state to the Operational state. Hence, the peer which transits to Operational state sends a Probe Operational message to alter the state of the corresponding peer to Operational state. At this point both peers are ready to resume the communication. Taking into account this behavior, Eq. (3) presents the value of $T_{recovery}$ for this scenario.

$$T_{recovery} = T_{RTx} + R_{TT} + \gamma + T_{send} + min(\tau_{\mathrm{A}}, \tau_B) \tag{3}$$

We define τ_{A} (τ_B) as the time elapsed between the sending of the first packet lost (in any node) and the starting time of the Send Timer on A (B). In the case considered, we are interested in the time at which the first node detecting the failure starts its Send Timer (in Fig. 5a, node A), condition that is expressed in general as $min(\tau_{\mathrm{A}}, \tau_{\mathrm{B}})$. Then after T_{send} time (the value of the Send Timer), node A sends a Probe Exploring that is never returned, so a new path is explored after T_{RTx} (the value of the Retransmission Timer) seconds. The new path is explored by sending a Probe Exploring message, requiring $\gamma_{\mathrm{AB}} + \gamma_{\mathrm{BA}} + \gamma_{\mathrm{AB}}$, or in other words, $R_{TT} + \gamma_{\mathrm{AB}}$. In order to make Eq. (3) independent from the actual node detecting the failure, we express $R_{TT} + \gamma_{\mathrm{AB}}$ in Eq. (3) as $R_{TT} + \gamma$, being γ the end-to-end delay from the node discovering the failure to its peer.



(a) Probe Exploring sent by A arrives at B before B enters into Exploration State.

(b) Both peers are in Exploring State when a valid path is discovered

Fig. 5. Path exploration transitions: Bidirectional traffic, Two-Way failure.

2. The second case corresponds to Fig. 5a (dashed lines). In this case, node A reaches the Exploring state before node B. The first Probe Exploring message sent by B is lost since this message is sent through the failed path. While node B is in the Exploring state, node A sends a Probe Exploring message that reaches node B, changing its state to Inbound_OK. In this transition, node B sends a Probe Inbound_OK message. When this message reaches node A, A changes its state to Operational, sending a Probe Operational message, which after reception, changes the state of node B to Operational. At this stage both peers are ready to resume the communication. This state sequence yields to the same equation as the first case (Eq. (3)).
3. The third case corresponds to Fig. 5b. In this case, peer B reaches the Exploring state prior to receiving the Probe Exploring message from A. In this scenario both peers perform the transition Exploring→Inbound_OK→Operational and are able to resume the communication once a Probe Inbound_OK is received (Eq. (4)).

$$T_{recovery} = T_{RTx} + R_{TT} + T_{send} + max(\tau_{\mathrm{A}}, \tau_B) \tag{4}$$

Now, the time at which the process is finished is driven by the last node detecting the failure.

4.2. Bidirectional traffic, One-Way failure

Fig. 6 presents the only possible state sequence for path exploration in a scenario where bidirectional traffic is affected by an outage on any unidirectional path (in this case, from A to B). As the Send Timer is set each time a packet is sent, and stopped each time a packet is received, the Send Timer in peer A never expires since the path from B to A is not affected by the outage. On the other hand, due to the outage, B does not receive packets from A so its Send Timer expires triggering a transition to the Exploring State. Then B sends a Probe Exploring message, which is received by A, and A transits to the Inbound_OK state. This Probe Exploring message reaches A using the current path, since the path from B to A is not affected by the outage. In this transition, A sends a Probe Inbound_OK to B, using the current path, which is lost. A new Probe Inbound_OK message sent through other path succeeds in arriving to B. As a consequence, B moves to the Operational state again, sending an Operational Probe to A. Once received the Probe Operational, A transits to Operational. Eq. (5) presents the value of $T_{recovery}$ for this scenario.

$$T_{recovery} = T_{RTx} + R_{TT} + \gamma_{\mathrm{BA}} + T_{send} + \tau_B \tag{5}$$

Note that the state transition sequence presented above is the only possible one, since the Send Timer can expire only in one of the nodes.

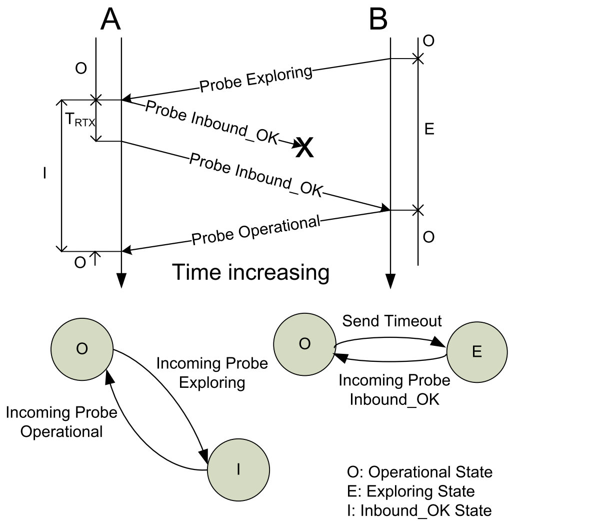


Fig. 6. Path exploration transitions: Bidirectional traffic, One-Way failure.

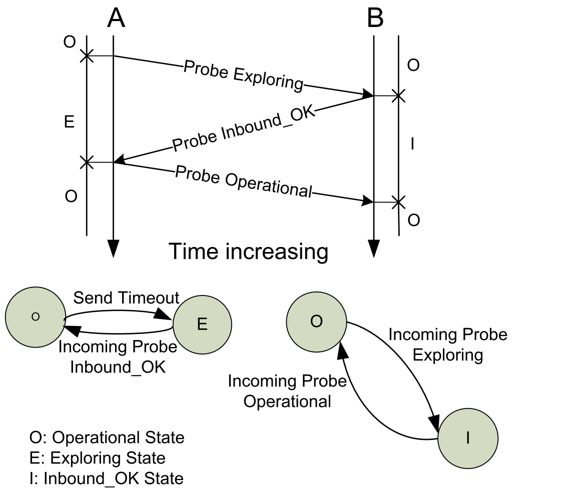


Fig. 7. Path exploration transitions: Unidirectional traffic, Two-Way failure.

4.3. Generic case for Bidirectional traffic

Given the equations for $T_{recovery}$ presented in the previous subsections, we now provide a general expression to be used for Bidirectional traffic, regardless the type of failure.

$$T_{recovery} = T_{RTx} + R_{TT} + T_{send} + min(\tau + \gamma, \tau_c) \tag{6}$$

In Eq. (6), τ corresponds to the peer whose Send Timer expires first ($min(\tau_A, \tau_B)$). γ is the end-to-end delay between the node whose Send Timer expires first and the peer. τ_c is the τ of the node whose Send Timer expires the last (i.e. $\tau_c = max(\tau_A, \tau_B)$). The case $\tau + \gamma < \tau_c$ corresponds to the first Probe Exploring message sent through a valid path reaching the peer while it is still in the Operational state. The case $\tau + \gamma > \tau_c$ corresponds to the first Probe Exploring message reaching the peer when it is in the Exploring state. Finally it is worth to note that the Bidirectional traffic, One-Way failure case (Eq. (5)) is just a particular case of expression (6) in which $max(\tau_A, \tau_B) = \infty$.

4.4. Unidirectional traffic, Two-Way failure

In this scenario only one state sequence is possible. As the Send Timer is set up each time a packet is sent, the Send Timer is only running on the node sending the packets. Fig. 7 presents the state sequence and the messages exchanged for the path exploration mechanism when node A is the sending active peer. When the failure occurs, A stops receiving Keepalive messages and its Send Timer expires, changing its state to Exploring and sending a Probe Exploring message. Upon reception of the Probe Exploring message, node B modifies its state to Inbound_OK and sends a Probe Inbound_OK message to A. When this probe is received on A, its state changes to Operational and a Probe Operational message is sent to B. Once A reaches the Operational state again the application is ready to resume the communication. Eq. (7) presents the value of $T_{recovery}$ for this scenario.

$$T_{recovery} = T_{RTx} + R_{TT} + T_{send} + \tau_A \tag{7}$$

4.5. Unidirectional traffic, One-Way failure in the data path

Due to the characteristics of REAP, this scenario corresponds exactly to the same state machine transition sequence as in the previous subsection (Section 4.4). Note that the $T_{recovery}$ value in this case is different from the previous section, being τ the difference between both cases.

5. Characterization of τ

Most of the components of the expressions presented in Section 4 are simple to characterize. However, this is not the case for the τ parameter. In the following sections we provide a set of equations to characterize τ for each of the cases presented in Section 4. The results are upper bounds of τ that are always a *supremum* (or *least upper bound*, i.e. the smallest real number that is greater than or equal to every possible τ). We use the term *maximal* for the case in which τ equals the *upper bound*, leaving the term *supremum* for the case in which τ never reaches the upper bound.

5.1. Bidirectional traffic, Two-Way failure

The approach followed for characterizing τ is to identify the only four cases in which the maximum value for τ can occur regardless the starting times for sending packets at A and B and the time of the failure:

- Case π: The first packet lost was sent by A, and τ_A reaches its maximum value ($\tau_{A\pi}$).
- Case θ: The first packet lost was sent by B, and τ_B reaches its maximum value ($\tau_{B\theta}$).
- Case ρ: The first packet lost was sent by A, and τ_B reaches its maximum value ($\tau_{B\rho}$).
- Case σ: The first packet lost was sent by B, and τ_A reaches its maximum value ($\tau_{A\sigma}$).

Taking into account the values of τ_A and τ_B for each of the scenarios described above, the maximum value of τ is:

$$\tau_{max} = max\left[min(\tau_{A\pi}, \tau_{B\pi}), min(\tau_{B\theta}, \tau_{A\theta}), min(\tau_{B\rho}, \tau_{A\rho}), min(\tau_{A\sigma}, \tau_{B\sigma})\right] \tag{8}$$

Note that cases $\pi - \theta$ and $\rho - \sigma$ are symmetric but we need to keep them as different cases because they depend on characteristics ($\alpha/\alpha', \Delta_A/\Delta_B, \gamma_{AB}/\gamma_{BA}$) that can be different in the two sides of the bidirectional communication, but that have to be computed simultaneously. In the following subsections we analyze in depth cases π and ρ, and present the final (symmetric) expressions for θ and σ, for which Δ_A, Δ_B, α, γ_{AB} and γ_{BA} are exchanged respectively by Δ_B, Δ_A, α', γ_{BA} and γ_{AB}. Next we prove that it is impossible to find a case, μ, in which $min(\tau_{A\mu}, \tau_{B\mu}) > \tau_{max}$, demonstrating that the maximum must occur in any of the four cases identified (π, ρ, θ, σ). A more general expression, independent from the specific location of the point of failure, which is an upper bound of τ (τ_{upp}) is provided in Section 6.

5.1.1. Case π

We first identify the case in which the maximum delay between the loss of the first packet sent by A and the start of the Send Timer on A occurs. Then we analyze the values of τ_A and τ_B for this specific scenario.

The worst case scenario in this situation (Fig. 8), that depends on the timing of the packets sent at both nodes, corresponds to the exchange of packets which leads to the greatest difference between T_{recA} (time at which the last packet sent by B arrives at A) and T_{lostA} (sending time of the first packet lost on A). This difference achieves its maximum when T_{lostA} is the lowest possible value and T_{recA} to its highest value. T_{send} will be started in A when the next packet is sent by A after the last packet from B was received (T_{recA}, see Fig. 8). Considering that A sends a new packet each $\varDelta_A$ seconds, the value of $\tau_{A\pi}$ can be expressed using the ceil function as $\left\lceil \frac{T_{recA} - T_{lostA}}{\varDelta_A} \right\rceil \Delta_A$.

In order for T_{recA} to be the highest value, the last packet which arrived correctly to peer A must be sent at the latest possible time, hence T_{lostB} approaches to the highest limit imposed in Eq. (2). In the same way, in order for T_{lostA} to be the lowest value, it must reach the lowest limit imposed by Eq. (1). Following the reasoning presented above, Eq. (9) presents the values for T_{lostA}, T_{lostB} and T_{recA}, with $\epsilon \to 0$, representing the packet sent at B just ϵ seconds before the failure could drop the packet.

$$\begin{aligned} T_{lostA} &= T_{fail} - \alpha\gamma_{AB} \\ T_{lostB} &= T_{fail} - \alpha'\gamma_{BA} + \varDelta_B - \epsilon \\ T_{recA} &= T_{lostB} + \gamma_{BA} - \varDelta_B \end{aligned} \tag{9}$$

Therefore

$$T_{recA} - T_{lostA} = (1 - \alpha')\gamma_{BA} + \alpha\gamma_{AB} - \epsilon$$

Note that as ϵ tends to zero, its contribution is irrelevant inside the ceil approximation, hence $\tau_{A\pi}$ is a *maximal* of τ.

$$\tau_{A\pi} = \left\lceil \frac{(1 - \alpha')\gamma_{BA} + \alpha\gamma_{AB}}{\varDelta_A} \right\rceil \Delta_A \tag{10}$$

We now characterize the corresponding value of τ in peer B ($\tau_{B\pi}$). The Send Timer in peer B is started on the first packet sent after the reception of the last packet from peer A (T_{recB}, see Figs. 9 and 10). By definition, $\tau_{B\pi}$ is

$$\tau_{B\pi} \leqslant T_{sendB} - T_{lostA} \tag{11}$$

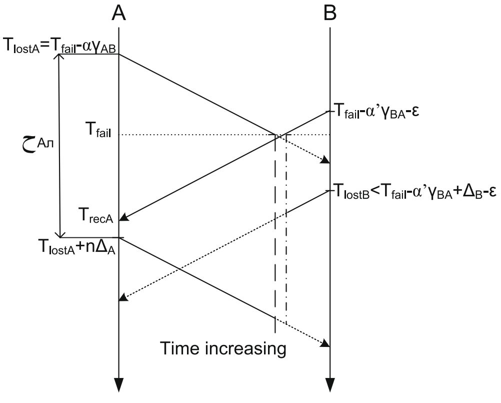


Fig. 8. Maximum $\tau_{A\pi}$ for first packet lost sent by A.

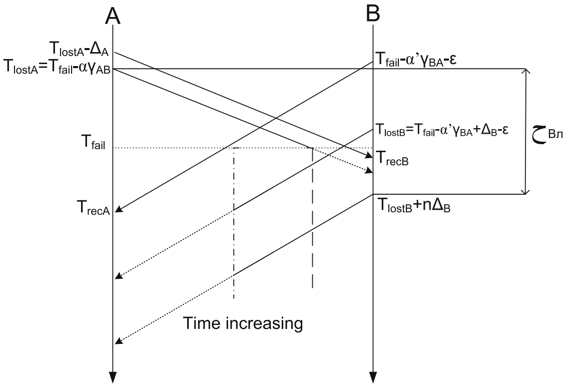


Fig. 9. Value of $\tau_{B\pi}$ when $T_{lostB} \leqslant T_{recB}$ for maximum τ_A and first packet lost sent by A.

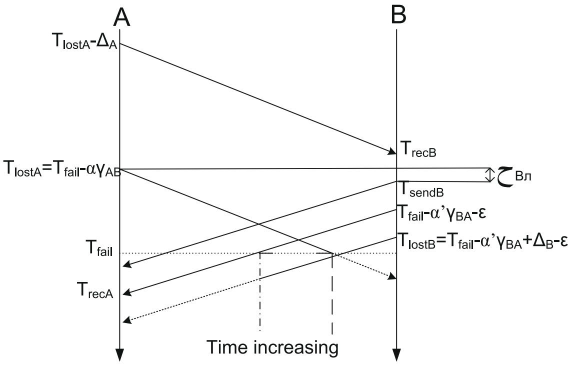


Fig. 10. Value of $\tau_{B\pi}$ when $T_{lostB} > T_{recB}$ for maximum τ_A and first packet lost sent by A.

where T_{sendB} is the sending time of the packet starting the Send Timer at B.

Depending on the packet timing and the end-to-end delays, two situations could occur: (i) $T_{lostB} \leqslant T_{recB}$ and (ii) $T_{lostB} > T_{recB}$.

For $T_{lostB} \leqslant T_{recB}$ (see Fig. 9), Eq. (12) presents the relation between the end-to-end delays, α, and the packet timing of each node, which makes $T_{lostB} < T_{recB}$.

$$\begin{aligned} T_{recB} &= T_{lostA} + \gamma_{AB} - \varDelta_A \\ T_{recB} &= T_{fail} + (1 - \alpha)\gamma_{AB} - \varDelta_A \\ T_{recB} &\geqslant T_{lostB} \Rightarrow (1 - \alpha)\gamma_{AB} + \alpha'\gamma_{BA} \geqslant \Delta_A + \varDelta_B \end{aligned} \tag{12}$$

Note that the values of T_{lostA} and T_{lostB} are the same of Eq. (9). To calculate the moment at which the Send Timer is set on peer B, we consider two steps. First the difference between T_{recB} and T_{lostB} is calculated. The next packet sent by B after T_{recB} is the packet setting the Send Timer. In this way we know the instant at which the Send Timer is started after T_{lostB}. In the second step, the distance between T_{lostB} and T_{lostA} is calculated. Adding these two values we obtain $\tau_{B\pi}$.

$$\tau_{B\pi} < \left\lceil \frac{T_{recB} - T_{lostB}}{\varDelta_B} \right\rceil \Delta_B + (T_{lostB} - T_{lostA}) \tag{13}$$

From the T_{recB} expression presented at (12), we obtain:

$$\left\lceil \frac{T_{recB} - T_{lostB}}{\varDelta_B} \right\rceil \Delta_B = \left\lceil \frac{(1 - \alpha)\gamma_{AB} + \alpha'\gamma_{BA} - \varDelta_A - \varDelta_B}{\varDelta_B} \right\rceil \Delta_B \tag{14}$$

$T_{lostB} - T_{lostA}$ can be computed as follows.

$$T_{lostB} - T_{lostA} = \alpha\gamma_{AB} + \alpha'\gamma_{BA} + \varDelta_B - \epsilon \tag{15}$$

Note that $T_{lost\mathrm{B}} > T_{lost\mathrm{A}}$ since we have defined in the scenario that the first packet lost was sent by A. Combining expressions (13)–(15) in Eq. (16),

$$\tau_{\mathrm{B}\pi} = \alpha\gamma_{\mathrm{AB}} - \alpha'\gamma_{\mathrm{BA}} + \Delta_{\mathrm{B}} + \left\lceil \frac{(1-\alpha)\gamma_{\mathrm{AB}} + \alpha'\gamma_{\mathrm{BA}} - \Delta_{\mathrm{A}} - \Delta_{\mathrm{B}}}{\Delta_{\mathrm{B}}} \right\rceil \Delta_{\mathrm{B}} \tag{16}$$

Now we solve Eq. (11) for the case $T_{lost\mathrm{B}} > T_{rec\mathrm{B}}$ (see Fig. 10). $T_{send\mathrm{B}}$ corresponds to the sending time of the next packet sent by B after receiving the last packet from A ($T_{rec\mathrm{B}}$). Eqs. (17) and (18) present the value of $T_{lost\mathrm{B}}$ and $T_{lost\mathrm{A}}$ for this case.

$$T_{lost\mathrm{B}} = T_{fail} - \alpha'\gamma_{\mathrm{BA}} + \Delta_{\mathrm{B}} - \epsilon \tag{17}$$

$$T_{lost\mathrm{A}} = T_{fail} - \alpha\gamma_{\mathrm{AB}} \tag{18}$$

The distance between $T_{lost\mathrm{B}}$ and $T_{rec\mathrm{B}}$ can be calculated taking into account the values of Eqs. (17) and (18).

$$T_{rec\mathrm{B}} = T_{lost\mathrm{A}} + \gamma_{\mathrm{AB}} - \Delta_{\mathrm{A}}$$

$$T_{rec\mathrm{B}} = T_{fail} + (1-\alpha)\gamma_{\mathrm{AB}} - \Delta_{\mathrm{A}}$$

$$T_{lost\mathrm{B}} - T_{rec\mathrm{B}} = -\alpha'\gamma_{\mathrm{BA}} - (1-\alpha)\gamma_{\mathrm{AB}} + \Delta_{\mathrm{A}} + \Delta_{\mathrm{B}} - \epsilon \tag{19}$$

$T_{send\mathrm{B}}$ can be obtained by considering the number of inter-packet intervals at B (of duration Δ_{B}) that fit into the distance between $T_{lost\mathrm{B}}$ and $T_{rec\mathrm{B}}$. The value of $T_{send\mathrm{B}}$ is presented in Eq. (20).

$$T_{send\mathrm{B}} = T_{lost\mathrm{B}} - \left\lfloor \frac{-\alpha'\gamma_{\mathrm{BA}} - (1-\alpha)\gamma_{\mathrm{AB}} + \Delta_{\mathrm{A}} + \Delta_{\mathrm{B}} - \epsilon}{\Delta_{\mathrm{B}}} \right\rfloor \Delta_{\mathrm{B}} \tag{20}$$

Combining Eqs. (17) and (18) the relationship between $T_{lost\mathrm{A}}$ and $T_{lost\mathrm{B}}$ is

$$T_{lost\mathrm{B}} - T_{lost\mathrm{A}} = \alpha\gamma_{\mathrm{AB}} - \alpha'\gamma_{\mathrm{BA}} + \Delta_{\mathrm{B}} - \epsilon \tag{21}$$

Including Eqs. (20) and (21) into Eq. (11), we can find a *supremum* (due to the presence of the ϵ) to the value of τ_{B} as shown in Eq. (22).

$$\tau_{\mathrm{B}\pi} = \alpha\gamma_{\mathrm{AB}} - \alpha'\gamma_{\mathrm{BA}} + \Delta_{\mathrm{B}} - \left\lfloor \frac{-\alpha'\gamma_{\mathrm{BA}} - (1-\alpha)\gamma_{\mathrm{AB}} + \Delta_{\mathrm{A}} + \Delta_{\mathrm{B}}}{\Delta_{\mathrm{B}}} \right\rfloor \Delta_{\mathrm{B}} \tag{22}$$

A summary of the values for $\tau_{\mathrm{B}\pi}$ is provided next:

- if $(1-\alpha)\gamma_{\mathrm{AB}} + \alpha'\gamma_{\mathrm{BA}} \geqslant \Delta_{\mathrm{A}} + \Delta_{\mathrm{B}}$

$$\tau_{\mathrm{B}\pi} = \alpha\gamma_{\mathrm{AB}} - \alpha'\gamma_{\mathrm{BA}} + \Delta_{\mathrm{B}} + \left\lceil \frac{(1-\alpha)\gamma_{\mathrm{AB}} + \alpha'\gamma_{\mathrm{BA}} - \Delta_{\mathrm{A}} - \Delta_{\mathrm{B}}}{\Delta_{\mathrm{B}}} \right\rceil \Delta_{\mathrm{B}}$$

- if $(1-\alpha)\gamma_{\mathrm{AB}} + \alpha'\gamma_{\mathrm{BA}} < \Delta_{\mathrm{A}} + \Delta_{\mathrm{B}}$

$$\tau_{\mathrm{B}\pi} = \alpha\gamma_{\mathrm{AB}} - \alpha'\gamma_{\mathrm{BA}} + \Delta_{\mathrm{B}} - \left\lfloor \frac{-\alpha'\gamma_{\mathrm{BA}} - (1-\alpha)\gamma_{\mathrm{AB}} + \Delta_{\mathrm{A}} + \Delta_{\mathrm{B}}}{\Delta_{\mathrm{B}}} \right\rfloor \Delta_{\mathrm{B}} \tag{23}$$

being $\tau_{\mathrm{B}\pi}$ a *supremum* of τ. Note that for the second case, $\tau_{\mathrm{B}} < \tau_{\mathrm{B}\pi} < \tau_{\mathrm{B}} + \Delta_{\mathrm{B}}$.

5.1.2. Case θ

Due to the symmetry of the system, τ on B when the first packet lost is sent by B corresponds to the same scenario as $\tau_{\mathrm{A}\pi}$ swapping node A by node B. The equations defining $\tau_{\mathrm{B}\theta}$ can be obtained following the procedure used in Section 5.1.1 after exchanging Δ_{A} by Δ_{B}, Δ_{B} by Δ_{A}, α by α', γ_{AB} by γ_{BA} and γ_{BA} by γ_{AB} in Eqs. (9) and (10):

$$\tau_{\mathrm{B}\theta} = \left\lceil \frac{(1-\alpha)\gamma_{\mathrm{AB}} + \alpha'\gamma_{\mathrm{BA}}}{\Delta_{\mathrm{B}}} \right\rceil \Delta_{\mathrm{B}} \tag{24}$$

As in the case of $\tau_{\mathrm{A}\pi}$, $\tau_{\mathrm{B}\theta}$ is a *maximal* of τ.

Correspondingly, $\tau_{\mathrm{A}\theta}$ is symmetric to $\tau_{\mathrm{B}\pi}$, so

- if $(1-\alpha')\gamma_{\mathrm{BA}} + \alpha\gamma_{\mathrm{AB}} \geqslant \Delta_{\mathrm{A}} + \Delta_{\mathrm{B}}$

$$\tau_{\mathrm{A}\theta} = \alpha'\gamma_{\mathrm{BA}} - \alpha\gamma_{\mathrm{AB}} + \Delta_{\mathrm{A}} + \left\lceil \frac{(1-\alpha')\gamma_{\mathrm{BA}} + \alpha\gamma_{\mathrm{AB}} - \Delta_{\mathrm{A}} - \Delta_{\mathrm{B}}}{\Delta_{\mathrm{A}}} \right\rceil \Delta_{\mathrm{A}}$$

- if $(1-\alpha')\gamma_{\mathrm{BA}} + \alpha\gamma_{\mathrm{AB}} < \Delta_{\mathrm{A}} + \Delta_{\mathrm{B}}$

$$\tau_{\mathrm{A}\theta} = \alpha'\gamma_{\mathrm{BA}} - \alpha\gamma_{\mathrm{AB}} + \Delta_{\mathrm{A}} - \left\lfloor \frac{-\alpha\gamma_{\mathrm{AB}} - (1-\alpha')\gamma_{\mathrm{BA}} + \Delta_{\mathrm{A}} + \Delta_{\mathrm{B}}}{\Delta_{\mathrm{A}}} \right\rfloor \Delta_{\mathrm{A}} \tag{25}$$

As in the case of $\tau_{\mathrm{B}\pi}$, $\tau_{\mathrm{A}\theta}$ is a *supremum* of $\tau_{\mathrm{A}}(\tau_{\mathrm{A}} < \tau_{\mathrm{A}\theta} < \tau_{\mathrm{A}} + \Delta_{\mathrm{A}})$.

5.1.3. Case ρ

In this section, we discuss the situation in which the maximum delay between the loss of the first packet sent by A and the start of the Send Timer on B occurs. Then we analyze the values of τ_{A} and τ_{B} for this specific scenario. Fig. 11 presents the worst case scenario for setting the Send Timer on B ($\tau_{\mathrm{B}\rho}$) when the first lost packet corresponds to A. The time at which B receives the last packet from A ($T_{rec\mathrm{B}}$) is stated in Eq. (26).

$$T_{rec\mathrm{B}} = T_{lost\mathrm{A}} + \gamma_{\mathrm{AB}} - \Delta_{\mathrm{A}} \tag{26}$$

The time when the Send Timer is set on B corresponds to the next packet sent after receiving the last packet from A. Regardless of the sending time of the last packet from A, the worst possible case occurs when B sends a packet Δ_{B} seconds after $T_{rec\mathrm{B}}$, being this time $T_{send\mathrm{B}}$. The value of τ for this case, that is a *maximal*, is presented in Eq. (27).

$$\tau_{\mathrm{B}\rho} = T_{send\mathrm{B}} - T_{lost\mathrm{A}} = T_{rec\mathrm{B}} + \Delta_{\mathrm{B}} - T_{lost\mathrm{A}}$$

$$\tau_{\mathrm{B}\rho} = \gamma_{\mathrm{AB}} - \Delta_{\mathrm{A}} + \Delta_{\mathrm{B}}. \tag{27}$$

Note that the worst possible case of τ_{B} in this scenario corresponds to the arrival of the last packet successfully sent by A just after the sending of a packet by B.

In Fig. 12 we detail the relevant parameters for this case on node A. The Send Timer on A is started when A sends the next packet after the last packet sent from B to A arrives ($T_{rec\mathrm{A}}$). Therefore

$$\tau_{\mathrm{A}\rho} = \left\lceil \frac{T_{rec\mathrm{A}} - T_{lost\mathrm{A}}}{\Delta_{\mathrm{A}}} \right\rceil \Delta_{\mathrm{A}} \tag{28}$$

We estimate the time at which the last packet from B arrives ($T_{rec\mathrm{A}}$) considering the time at which this packet was sent, $T_{s\mathrm{B}}$, plus the end-to-end delay from B to A.

$$T_{rec\mathrm{A}} = T_{s\mathrm{B}} + \gamma_{\mathrm{BA}} \tag{29}$$

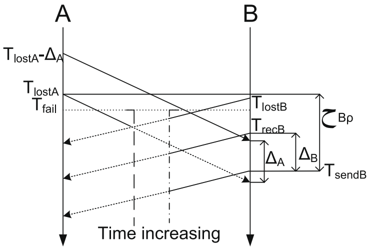


Fig. 11. Maximum $\tau_{\mathrm{B}\rho}$ for first packet lost sent by A.

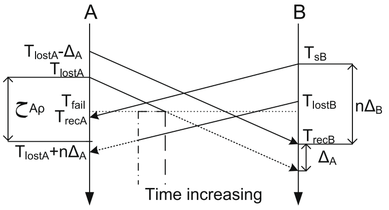


Fig. 12. Value of $\tau_{\mathrm{A}\rho}$ for maximum τ_{B} and first packet lost sent by A.

We can set a relation between $T_{s\mathrm{B}}$ and $T_{lost\mathrm{A}}$ by analyzing the relationship among the time at which the last packet received at B arrived ($T_{rec\mathrm{B}}$) and $T_{s\mathrm{B}}$ through a new positive integer parameter n.

$$T_{s\mathrm{B}} = T_{lost\mathrm{A}} + \gamma_{\mathrm{AB}} - \varDelta_{\mathrm{A}} - n\Delta_{\mathrm{B}} \tag{30}$$

To find the value of n we know that $T_{s\mathrm{B}}$ must be the greatest possible value lower than $T_{lost\mathrm{B}}$ in order to be received on A ($T_{s\mathrm{B}} < T_{fail} - \alpha'\gamma_{\mathrm{BA}}$).

$$T_{lost\mathrm{A}} + \gamma_{\mathrm{AB}} - \varDelta_{\mathrm{A}} - n\Delta_{\mathrm{B}} < T_{fail} - \alpha'\gamma_{\mathrm{BA}}$$
$$T_{lost\mathrm{A}} = T_{fail} - \alpha\gamma_{\mathrm{AB}}$$
$$n\Delta_{\mathrm{B}} > \alpha'\gamma_{\mathrm{BA}} + (1-\alpha)\gamma_{\mathrm{AB}} - \varDelta_{\mathrm{A}}$$
$$n = \left\lceil \frac{\alpha'\gamma_{\mathrm{BA}} + (1-\alpha)\gamma_{\mathrm{AB}} - \varDelta_{\mathrm{A}}}{\varDelta_{\mathrm{B}}} \right\rceil \tag{31}$$

Eq. (32) shows the time at which the first packet sent from B is received on A and combines Eqs. (28)–(31) to find the final value for $\tau_{\mathrm{A}\rho}$.

$$\tau_{\mathrm{A}\rho} = \left\lceil \frac{R_{TT} - \varDelta_{\mathrm{A}} - \left\lceil \frac{\alpha'\gamma_{\mathrm{BA}} + (1-\alpha)\gamma_{\mathrm{AB}} - \varDelta_{\mathrm{A}}}{\varDelta_{\mathrm{B}}} \right\rceil \Delta_{\mathrm{B}}}{\varDelta_{\mathrm{A}}} \right\rceil \varDelta_{\mathrm{A}} \tag{32}$$

being $\tau_{\mathrm{A}\rho}$ a *maximal* of τ.

5.1.4. Case σ

Due to symmetry considerations, $\tau_{\mathrm{A}\sigma}$ corresponds to the same scenario as $\tau_{\mathrm{B}\rho}$, after swapping node A by node B. The equations can be obtained following the same procedure used in Section 5.1.3, i.e. exchanging $\varDelta_{\mathrm{A}}$ by $\varDelta_{\mathrm{B}}$, $\varDelta_{\mathrm{A}}$ by $\varDelta_{\mathrm{B}}$, α by α', γ_{AB} by γ_{BA} and γ_{BA} by γ_{AB} in Eq. (26).

$$\tau_{\mathrm{A}\sigma} = \gamma_{\mathrm{BA}} - \varDelta_{\mathrm{B}} + \varDelta_{\mathrm{A}} \tag{33}$$

being $\tau_{\mathrm{A}\sigma}$ a *maximal* of τ. The corresponding case in B ($\tau_{\mathrm{B}\sigma}$) is

$$\tau_{\mathrm{B}\sigma} = \left\lceil \frac{R_{TT} - \varDelta_{\mathrm{B}} - \left\lceil \frac{\alpha\gamma_{\mathrm{AB}} + (1-\alpha')\gamma_{\mathrm{BA}} - \varDelta_{\mathrm{B}}}{\varDelta_{\mathrm{A}}} \right\rceil \Delta_{\mathrm{A}}}{\varDelta_{\mathrm{B}}} \right\rceil \varDelta_{\mathrm{B}}$$

being $\tau_{\mathrm{B}\sigma}$ a *maximal* of τ.

5.1.5. Proof of the Maximality of τ_{max}

For a given case (two nodes communicating, and a failure among them), $T_{recovery}$ depends on the minimum value of τ in side A and τ in side B. Therefore, to obtain the worst case for $T_{recovery}$, we should look for the worst case, i.e. the maximum case, of this $min(\tau_{\mathrm{A}}, \tau_{\mathrm{B}})$. Eq. (8) assumes that this maximum of τ happens in one of four particular cases. These cases are defined by considering the situation at which τ is maximum for one side, and which is the side sending the first packet lost.

However, it could be thought that other cases different than these could lead to a greater value of $min(\tau_{\mathrm{A}}, \tau_{\mathrm{B}})$, since the assumptions made to build the four cases (when a maximum value of tau in one side occurs) did not stressed the maximality of $min(\tau_{\mathrm{A}}, \tau_{\mathrm{B}})$.

In the following paragraphs we prove that τ_{max} (Eq. (8)) provides a maximum for all possible combinations of τ_{A} and τ_{B}, i.e. $\forall\langle\tau_{\mathrm{A}\mu}, \tau_{\mathrm{B}\mu}\rangle, min(\tau_{\mathrm{A}\mu}, \tau_{\mathrm{B}\mu}) \leqslant \tau_{max}$. This means that we prove that the maximum occurs in any of the specific cases (π, ρ, θ, σ). Due to the symmetry inherent to Eq. (8) we focus on proving, for the case in which the first packet lost is sent by A, that

$$\forall\tau_{\mathrm{A}\mu}, \tau_{\mathrm{B}\mu}/\tau = min(\tau_{\mathrm{A}\mu}, \tau_{\mathrm{B}\mu}) \Rightarrow \tau \leqslant max\left[min(\tau_{\mathrm{A}\pi}, \tau_{\mathrm{B}\pi}), min(\tau_{\mathrm{B}\rho}, \tau_{\mathrm{A}\rho})\right] \tag{34}$$

for every possible case of $\tau_{\mathrm{A}\mu}$ and $\tau_{\mathrm{B}\mu}$ when the first packet lost is sent by A. An analogous demonstration can be done for the cases when the first packet lost is sent by B.

To prove Eq. (34), we consider the four possible combinations of the values of $\tau_{\mathrm{A}\pi}$, $\tau_{\mathrm{B}\pi}$, $\tau_{\mathrm{A}\rho}$ and $\tau_{\mathrm{B}\rho}$:

1. $\tau_{\mathrm{A}\pi} < \tau_{\mathrm{B}\pi}$ and $\tau_{\mathrm{A}\rho} < \tau_{\mathrm{B}\rho} \Rightarrow \tau \leqslant max(\tau_{\mathrm{A}\pi}, \tau_{\mathrm{A}\rho})$
2. $\tau_{\mathrm{B}\pi} < \tau_{\mathrm{A}\pi}$ and $\tau_{\mathrm{B}\rho} < \tau_{\mathrm{A}\rho} \Rightarrow \tau \leqslant max(\tau_{\mathrm{B}\pi}, \tau_{\mathrm{B}\rho})$
3. $\tau_{\mathrm{A}\pi} < \tau_{\mathrm{B}\pi}$ and $\tau_{\mathrm{B}\rho} < \tau_{\mathrm{A}\rho} \Rightarrow \tau \leqslant max(\tau_{\mathrm{B}\rho}, \tau_{\mathrm{A}\pi})$
4. $\tau_{\mathrm{B}\pi} < \tau_{\mathrm{A}\pi}$ and $\tau_{\mathrm{A}\rho} < \tau_{\mathrm{B}\rho} \Rightarrow \tau \leqslant max(\tau_{\mathrm{B}\pi}, \tau_{\mathrm{A}\rho})$

Note that, as explained in Section 5.1, by definition $\tau_{\mathrm{A}\pi} > \tau_{\mathrm{A}\rho}$ and $\tau_{\mathrm{B}\rho} > \tau_{\mathrm{B}\pi}$, since $\tau_{\mathrm{A}\pi}$ and $\tau_{\mathrm{B}\rho}$ are the worst possible cases for τ_{A} and τ_{B} respectively.

Consider the combination 1. We have to show that $\forall\tau, \tau \leqslant max(\tau_{\mathrm{A}\pi}, \tau_{\mathrm{A}\rho})$. As $\tau_{A\pi} > \tau_{\mathrm{A}\rho}$ the maximum is $\tau_{\mathrm{A}\pi}$, it should be proved that $\forall\tau_{\mathrm{A}\mu}, \tau_{\mathrm{B}\mu}, \tau = min(\tau_{\mathrm{A}\mu}, \tau_{\mathrm{B}\mu}) \leqslant \tau_{\mathrm{A}\pi}$. We know that $\forall\tau_{\mathrm{A}\mu}, \tau_{\mathrm{A}\mu} < \tau_{\mathrm{A}\pi}$. For every value of $\tau_{\mathrm{A}\mu}$ and $\tau_{\mathrm{B}\mu}, \tau = min(\tau_{\mathrm{A}\mu}, \tau_{\mathrm{B}\mu})$ at least is as small as $\tau_{\mathrm{A}\mu}$, and this is smaller than $\tau_{\mathrm{A}\pi}$, proving that $\tau \leqslant \tau_{\mathrm{A}\pi}$ for all possible combination of $\tau_{\mathrm{A}\mu}$ and $\tau_{\mathrm{B}\mu}$ with the constrains imposed by the first combination.

The same reasoning can be applied to the second case, combination 2, to show that, in this case, $\tau \leqslant max(\tau_{\mathrm{B}\pi}, \tau_{\mathrm{B}\rho}) = \tau_{\mathrm{B}\rho}$.

The third case (combination 3), imposes $\tau_{\mathrm{B}\rho} < \tau_{\mathrm{A}\rho}$ and $\tau_{\mathrm{A}\pi} < \tau_{\mathrm{B}\pi}$, hence $\tau_{\mathrm{A}\pi} < \tau_{\mathrm{A}\rho}$. Since $\tau_{\mathrm{A}\pi} > \tau_{\mathrm{A}\rho}$ by definition, this case is not possible.

Finally for the fourth case (combination 4), we have to show that for all combinations of $\tau_{\mathrm{A}\mu}$ and $\tau_{\mathrm{B}\mu}$, $\tau \leqslant max(\tau_{\mathrm{B}\pi}, \tau_{\mathrm{A}\rho})$. $\tau = min(\tau_{\mathrm{A}\mu}, \tau_{\mathrm{B}\mu})$ so we have to prove that there is not a value of $\tau_{\mathrm{A}\mu}$ and $\tau_{\mathrm{B}\mu}$ for which $min(\tau_{\mathrm{A}\mu}, \tau_{\mathrm{B}\mu}) > max(\tau_{\mathrm{B}\pi}, \tau_{\mathrm{A}\rho})$. This is equivalent to prove that the values of $\tau_{\mathrm{A}\mu}$ and $\tau_{\mathrm{B}\mu}$ are not within the intervals ($\tau_{\mathrm{B}\pi}$, $\tau_{\mathrm{B}\rho}$) and ($\tau_{\mathrm{A}\rho}$, $\tau_{\mathrm{A}\pi}$) at the same time. On the following lines we prove that this situation is not possible by showing that given $\tau_{\mathrm{A}\mu}$ within ($\tau_{\mathrm{A}\rho}$, $\tau_{\mathrm{A}\pi}$) there is not a value of $\tau_{\mathrm{B}\mu}$ greater than $\tau_{\mathrm{B}\pi}$.

Proof. Suppose an arbitrary value of $\tau_{\mathrm{A}\mu}$ and the corresponding $\tau_{\mathrm{B}\mu}$ value, for all possible values of $T_{lost\mathrm{A}}$ and $T_{lost\mathrm{B}}$. Eqs. (35) and (36) show the value of $\tau_{\mathrm{A}\mu}$ and $\tau_{\mathrm{B}\mu}$ calculated as in Section 5.1.1.

$$\tau_{\mathrm{A}\mu} = \left\lceil \frac{T_{lost\mathrm{B}} - T_{lost\mathrm{A}} + \gamma_{\mathrm{BA}} - \varDelta_{\mathrm{B}}}{\varDelta_{\mathrm{A}}} \right\rceil \Delta_{\mathrm{A}} \tag{35}$$

$$\tau_{\mathrm{B}\mu} = T_{lost\mathrm{B}} - T_{lost\mathrm{A}} + \left\lceil \frac{T_{lost\mathrm{B}} - T_{lost\mathrm{A}} + \gamma_{\mathrm{BA}} - \varDelta_{\mathrm{B}}}{\varDelta_{\mathrm{A}}} \right\rceil \Delta_{\mathrm{A}} \tag{36}$$

Now we find the constraints imposed by the range of possible values of τ_{A}.

- $\tau_{\mathrm{A}\mu} > \tau_{\mathrm{A}\rho}$

$$T_{lost\mathrm{A}} - T_{lost\mathrm{B}} + \gamma_{\mathrm{AB}} < \alpha'\gamma_{\mathrm{BA}} + (1-\alpha)\gamma_{\mathrm{AB}} - \varDelta_{\mathrm{B}} \tag{37}$$

- $\tau_{\mathrm{A}\mu} < \tau_{\mathrm{A}\pi}$

$$T_{lost\mathrm{B}} - T_{lost\mathrm{A}} < \alpha\gamma_{\mathrm{AB}} - \alpha'\gamma_{\mathrm{BA}} + \varDelta_{\mathrm{B}} \tag{38}$$

Supposing there is a $\tau_{\mathrm{B}\mu} > \tau_{\mathrm{B}\pi}$, then Eq. (39) must be true.

$$\begin{aligned} &T_{lost\mathrm{B}} - T_{lost\mathrm{A}} + \left\lceil \frac{T_{lost\mathrm{B}} - T_{lost\mathrm{A}} + \gamma_{\mathrm{BA}} - \varDelta_{\mathrm{B}}}{\varDelta_{\mathrm{A}}} \right\rceil \Delta_{\mathrm{A}} \\ &\quad > \alpha\gamma_{\mathrm{AB}} - \alpha'\gamma_{\mathrm{BA}} + \varDelta_{\mathrm{B}} + \left\lceil \frac{(1-\alpha)\gamma_{\mathrm{AB}} + \alpha'\gamma_{\mathrm{BA}} - \varDelta_{\mathrm{A}} - \varDelta_{\mathrm{B}}}{\varDelta_{\mathrm{B}}} \right\rceil \Delta_{\mathrm{B}} \end{aligned} \tag{39}$$

Imposing the constraints defined in Eqs. (37) and (38) into $\tau_{\mathrm{B}\pi}$ we obtain Eq. (40).

$$\tau_{\mathrm{B}\pi} > T_{lost\mathrm{B}} - T_{lost\mathrm{A}} + \left\lceil \frac{T_{lost\mathrm{B}} - T_{lost\mathrm{A}} + \gamma_{\mathrm{BA}} - \varDelta_{\mathrm{B}}}{\varDelta_{\mathrm{A}}} \right\rceil \Delta_{\mathrm{A}} \tag{40}$$

Eq. (40) combined with Eq. (36) imposes that $\tau_{\mathrm{B}\pi} > \tau_{\mathrm{B}\mu}$ so the condition $\tau_{\mathrm{B}\mu} > \tau_{\mathrm{B}\pi}$ cannot be fulfilled. This ends the proof. □

5.2. Bidirectional traffic, One-Way failure

In this section, we consider scenarios in which node A and B exchange bidirectional traffic and a failure occurs in only one of the directions of the communication. As it will be presented, this case is a particularization of the analysis performed for the Bidirectional traffic, Two-Way failure, in Section 5.1.

5.2.1. Failure in the path from A to B

Eq. (8), showed the expression which provides τ_{max} for a bidirectional traffic and Two-Way failure. This equation applies when the first packet lost was sent by either A or by B. In the current scenario, B does not suffer any packet loss since there is no failure in the path from B to A, hence $\tau_{\mathrm{B}\theta}$, $\tau_{\mathrm{A}\theta}$, $\tau_{\mathrm{A}\sigma}$ and $\tau_{\mathrm{B}\sigma}$ are not considered for this scenario. In the same way, equations in which the Send Timer is set on A are not considered, since A is always receiving packets and the Send Timer is being stopped and reset regularly. Therefore equations $\tau_{\mathrm{A}\pi}$, $\tau_{\mathrm{B}\pi}$ and $\tau_{\mathrm{A}\rho}$ are not considered. This reasoning yields to the fact that the τ_{max} for this scenario is equal to $\tau_{B\rho}$. Eq. (41) shows the equation to consider in this scenario.

$$\tau_{max} = \gamma_{\mathrm{AB}} - \varDelta_{\mathrm{A}} + \varDelta_{B} \tag{41}$$

being this value of τ a *maximal.*

5.2.2. Failure in the path from B to A

Following the same reasoning as in Section 5.2.1, for this scenario only $\tau_{\mathrm{A}\sigma}$ must be considered. Hence Eq. (42) shows the value of τ_{max} to consider in this scenario.

$$\tau_{max} = \gamma_{\mathrm{BA}} - \varDelta_{\mathrm{B}} + \varDelta_{\mathrm{A}} \tag{42}$$

being this value of τ a *maximal.*

5.3. Unidirectional traffic, Two-Way failure

Fig. 13 presents the worst case for a unidirectional traffic flow affected by a bidirectional failure. This case occurs when a Keepalive message is sent in the latest possible instant before the failure. Therefore, the activation of the Send Timer when the next packet is sent produces the highest delay on τ. It is important to note that the time at which the Keepalive message is sent depends on the time at which node B receives a packet, hence the Keepalive message sending time on B depends on the traffic of the peer. Eq. (43) presents the value of τ_{max} for this scenario.

$$\tau_{max} = \left\lceil \frac{T_{RKA} - T_{lost\mathrm{A}}}{\varDelta_{\mathrm{A}}} \right\rceil \Delta_{\mathrm{A}} \tag{43}$$

Due to the dependence between the Keepalive timer and the traffic sent by the peer, in order to calculate the value of τ_{max} we proceed in several stages. First we calculate the time at which peer A sends the packet which activates the Keepalive Timer on B (T_1, see Fig. 13). When the Keepalive timer expires on B, at time T_{SKA}, a Keepalive message is sent. T_{SKA} can be derived from T_1 by adding the end-to-end delay and the T_{KA}. Then we can obtain the relationship between T_{lostA} and T_{SKA}. Once the relation between T_{lostA} and T_{SKA} is found, the calculation of τ_{max} is done by finding the time at which the Keepalive message was received by A (T_{RKA}).

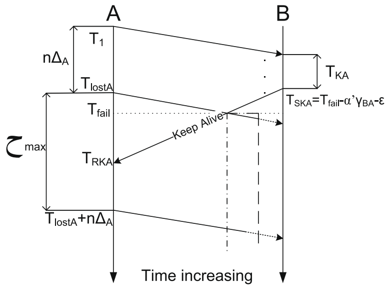


Fig. 13. Worst Case scenario for Unidirectional traffic affected by a Two-Way failure.

Eq. (44) presents these relationships.

$$\begin{aligned} T_{SKA} &= T_{fail} - \alpha'\gamma_{\mathrm{BA}} - \epsilon \\ T_1 &= T_{SKA} - T_{KA} - \gamma_{\mathrm{AB}} \\ T_{lost\mathrm{A}} &= T_1 + n\Delta_{\mathrm{A}} \end{aligned} \tag{44}$$

The time at which the first lost packet is sent by A (T_{lostA}) occurs a given positive integer number (n) of $\varDelta_{\mathrm{A}}$ periods after the packet that started the Keepalive at B was sent (T_1). Note that T_{lostA} is related with T_{fail} by the following equation:

$$T_{fail} - \alpha\gamma_{\mathrm{AB}} \leqslant T_{lost\mathrm{A}} < T_{fail} - \alpha\gamma_{\mathrm{AB}} + \varDelta_{\mathrm{A}} \tag{45}$$

Then

$$\begin{aligned} T_{lost\mathrm{A}} &\geqslant T_{fail} - \alpha\gamma_{\mathrm{AB}} \\ n\Delta_{\mathrm{A}} &\geqslant T_{fail} - \alpha\gamma_{\mathrm{AB}} - T_1 \\ n\Delta_{\mathrm{A}} &\geqslant (1-\alpha)\gamma_{\mathrm{AB}} + \alpha'\gamma_{\mathrm{BA}} + T_{KA} \end{aligned} \tag{46}$$

$$\begin{aligned} T_{lost\mathrm{A}} &< T_{fail} - \alpha\gamma_{\mathrm{AB}} + \varDelta_{\mathrm{A}} \\ n\Delta_{\mathrm{A}} &< T_{fail} - \alpha\gamma_{\mathrm{AB}} + \varDelta_{\mathrm{A}} - T_1 \\ n\Delta_{\mathrm{A}} &< (1-\alpha)\gamma_{\mathrm{AB}} + \alpha'\gamma_{\mathrm{BA}} + T_{KA} + \varDelta_{\mathrm{A}} \end{aligned} \tag{47}$$

Combining both equations, we obtain the value for $T_{lost\mathrm{A}}$.

$$T_{lost\mathrm{A}} = T_1 + \left\lfloor \frac{(1-\alpha)\gamma_{\mathrm{AB}} + \alpha'\gamma_{\mathrm{BA}} + T_{KA}}{\varDelta_{\mathrm{A}}} + 1 \right\rfloor \Delta_{\mathrm{A}} \tag{48}$$

The time at which the Keepalive is received is expressed as

$$T_{RK\mathrm{A}} = T_{SKA} + \gamma_{\mathrm{BA}} \tag{49}$$

τ_{max} corresponds to the difference between $T_{RK\mathrm{A}}$ and $T_{lost\mathrm{A}}$, taking into account that the Send Timer starts when the next packet is sent, as shown in Eq. (50).

$$\begin{aligned} T_{SKA} &= T_{fail} - \alpha'\gamma_{\mathrm{BA}} - \epsilon \\ T_{RKA} &= T_{SKA} + \gamma_{\mathrm{BA}} \\ T_1 &= T_{SKA} - T_{KA} - \gamma_{\mathrm{AB}} \\ T_{RKA} &= T_{fail} + (1-\alpha')\gamma_{\mathrm{BA}} - \epsilon \\ T_1 &= T_{fail} - \alpha'\gamma_{\mathrm{BA}} - \epsilon - T_{KA} - \gamma_{\mathrm{AB}} \\ \tau_{max} &= \left\lceil \frac{R_{TT} + T_{KA} - \lfloor \frac{(1-\alpha)\gamma_{\mathrm{AB}} + \alpha'\gamma_{\mathrm{BA}} + T_{KA}}{\varDelta_{\mathrm{A}}} + 1 \rfloor \Delta_{\mathrm{A}}}{\varDelta_{\mathrm{A}}} \right\rceil \Delta_{\mathrm{A}} \end{aligned} \tag{50}$$

This value of τ_{max} is a *maximal.*

5.4. Unidirectional traffic, One-Way failure in data path

Fig. 14 presents the worst case for a unidirectional traffic flow affected by a failure in the data path. This case occurs when the Keepalive Timer in peer B is started by the latest possible data packet sent by peer A. After a Keepalive Timer period, a Keepalive message is sent from B to A, resetting the Send Timer at A.

$$\begin{aligned} T_{RK\mathrm{A}} &= T_{fail} - \alpha\gamma_{\mathrm{AB}} + \gamma_{\mathrm{AB}} + T_{KA} + \gamma_{\mathrm{BA}} - \epsilon \\ T_{lost\mathrm{A}} &= T_{fail} - \alpha\gamma_{\mathrm{AB}} + \varDelta_{\mathrm{A}} - \epsilon \\ \tau_{max} &= \left\lceil \frac{T_{RK\mathrm{A}} - T_{lost\mathrm{A}}}{\varDelta_{\mathrm{A}}} \right\rceil \Delta_{\mathrm{A}} \end{aligned} \tag{51}$$

Eq. (52) shows the value of τ_{max} for this case, being this value of τ_{max} a *maximal.*

$$\tau = \left\lceil \frac{R_{TT} + T_{KA} - \varDelta_{\mathrm{A}}}{\varDelta_{\mathrm{A}}} \right\rceil \Delta_{\mathrm{A}} \tag{52}$$

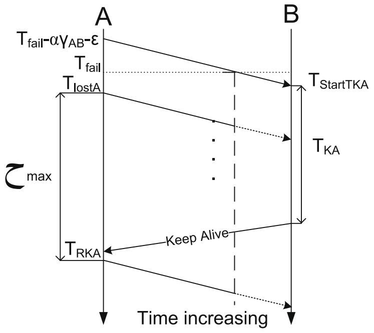


Fig. 14. Worst Case scenario for Unidirectional traffic affected by a failure in data path.

5.5. τ simulation results

In order to validate the results presented before, an extensive search of maximum values for τ has been performed. The objective of the simulation is to sample systematically the continuous space of starting times for a given communication scenario, to measure the corresponding value of τ, in order to check if the upper bounds obtained in our previous analysis hold.

Therefore, we set some values for the parameters that a given specific application *could find* in two particular nodes: Δ_A (and Δ_B for bidirectional communication), resulting from the own nature of the application; γ_{AB} and γ_{BA} resulting from the initial path used. Then we assume a type of failure, the details of this failure, i.e. the values of α and/or α' depending on the type of failure, that describe the failure position, and the time at which the failure occurs. The set of experiments is defined by setting different starting times for the periodic data transmission in A (B). With this, we aim to test all the different combinations of the defined starting times at A and B. When an experiment is defined (the communication to test, type of failure and starting times for the data exchanged), we use Matlab to simulate all the packets that would be exchanged among the communicating nodes, and as a result of this simulation, we obtain the value of τ for the experiment. The maximum value of τ for each experiment related with a communication scenario is the worst case value of τ for the communication, which is compared with the expected theoretical result, in particular, the results provided by equations:

- Eq. (8) for the Bidirectional traffic, Two-Way failure case.
- Eqs. (41) and (42) for the Bidirectional traffic, One-Way failure case.
- Eq. (50) for the Unidirectional traffic, Two-Way failure.
- Eq. (52) for the Unidirectional traffic, One-way failure in the data path respectively.

For each of the communication scenarios, the numerical parameters that define an experiment ($\Delta_A, \Delta_B, \gamma_{AB}, \gamma_{BA}$, α and α') have been generated using two approaches. In the first approach, we have generated a set of values for each parameter by defining a starting value that is incremented by a fixed step. Then, we have simulated all the configurations resulting from the combinations of these values (around 5500 samples for each Bidirectional traffic scenario and 1400 sample for each Unidirectional traffic scenario). In the second approach, we have generated randomly the values of the parameters (200 samples for each scenario), supposing a uniform distribution among fixed initial and final values with increments of 1 ms (Table 1). In order to obtain a more precise value for the maximum of τ, further simulations were performed with starting points close to the ones in which the higher τ value was obtained, using in this time smaller variations. Fig. 15 shows the percentage of error, for each sample, between the theoretical value of τ provided by our analysis and the value of τ measured in the simulator for the Bidirectional traffic, Two-Way failure scenario and 200 random samples. The configuration of the REAP timers is set as defined in the specification [3]. We only present these results due to length considerations, although the complete set of results[2] show similar behavior. In all cases, the difference between the theoretical analysis and the simulated model is 0 when the value of τ corresponds to a *maximal*, showing that the theoretical model is able to compute without error the value of τ_{max}. When the theoretical result is a *supremum*, the simulator provides results whose differences with the theoretical predictions can be made arbitrarily small by reducing the step used for the variation of the sending times at A and B.

Table 1
Traffic characteristics for random exploration.

Parameter	Initial value	Ending value
Δ_A	1 ms	200 ms
Δ_B	1 ms	200 ms
γ_{AB}	1 ms	100 ms
γ_{BA}	1 ms	100 ms
α	0	1
α'	0	1

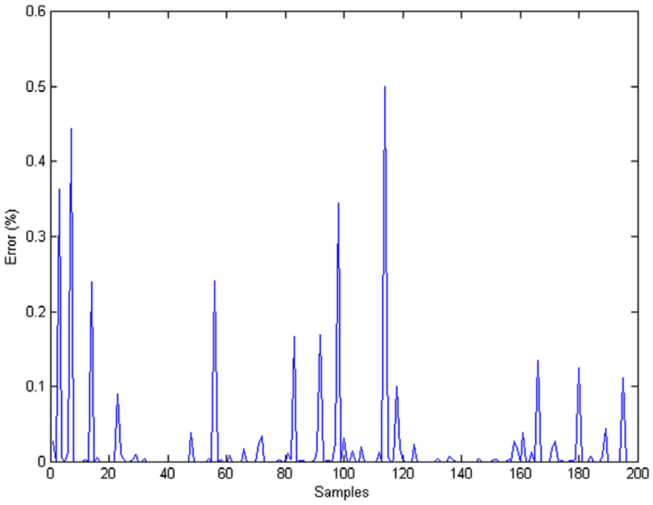


Fig. 15. Percentage of error between the theoretical and experimental results for 200 random samples, Bidirectional traffic, Two-Way failure case.

6. Upper bound for the recovery time regardless of the location and type of the failure

In this section we simplify the analytical results obtained in Section 5 to obtain an upper bound for $T_{recovery}$ independently of the failure point and type of failure for both the Bidirectional and Unidirectional traffic. This is the result expected to be useful for the configuration of REAP in a real deployment.

First an upper bound for τ_{max} regardless the point of failure is obtained for the scenarios that depended on the α and α' parameters, that were the Bidirectional traffic, Two-Way failure, and the Unidirectional traffic Two-Way failure. Then, we provide an upper bound for $T_{recovery}$ in the Bidirectional traffic and in the Unidirectional traffic scenarios independent from the failure type or location.

[2] The complete results data set can be obtained from http://enjambre.it.uc3m.es/aoliva/reap.html.

6.1. Upper bound for τ regardless the location of the failure: Bidirectional traffic, Two-Way failure

A *supremum* for τ for Bidirectional traffic, Two-Way failure, was provided in Eq. (8). In order to obtain a compact expression for an upper bound that does not depend on the location of the failure (i.e. on α or on α'), we first apply the inequalities $x + 1 \geqslant \lceil x \rceil$ and $x - 1 \leqslant \lfloor x \rfloor$ to obtain the following equation:

$$\begin{aligned} &max[min(\alpha\gamma_{\mathrm{AB}} + (1-\alpha')\gamma_{\mathrm{BA}} + \Delta_{\mathrm{A}}, \gamma_{\mathrm{AB}} + \Delta_{\mathrm{B}} - \Delta_{\mathrm{A}}),\\ &min(\alpha'\gamma_{\mathrm{BA}} + (1-\alpha)\gamma_{\mathrm{AB}} + \Delta_{\mathrm{B}}, \gamma_{\mathrm{BA}} + \Delta_{\mathrm{A}} - \Delta_{\mathrm{B}}),\\ &min(\gamma_{\mathrm{AB}} + \Delta_{\mathrm{B}} - \Delta_{\mathrm{A}}, R_{TT} - \alpha'\gamma_{\mathrm{BA}} - (1-\alpha)\gamma_{\mathrm{AB}} + \Delta_{\mathrm{A}}),\\ &min(\gamma_{\mathrm{BA}} + \Delta_{\mathrm{A}} - \Delta_{\mathrm{B}}, R_{TT} - \alpha\gamma_{\mathrm{AB}} - (1-\alpha')\gamma_{\mathrm{BA}} + \Delta_{\mathrm{B}})] \end{aligned} \quad (53)$$

In order to provide an upper bound, we maximize the values of each term in Eq. (53). For each term, we choose the values of α and α' maximizing it. For the cases π and σ, τ reaches its maximum value at node A so the values maximizing these terms are $\alpha = 1, \ \alpha' = 0$. For the cases θ and ρ, τ reaches its maximum at node B, hence the values maximizing these terms are $\alpha = 0, \ \alpha' = 1$. Therefore

$$\begin{aligned} \tau_{upp} = {} &max[min(R_{TT} + \Delta_{\mathrm{A}}, \gamma_{\mathrm{AB}} + \Delta_{\mathrm{B}} - \Delta_{\mathrm{A}}),\\ &min(R_{TT} + \Delta_{\mathrm{B}}, \gamma_{\mathrm{BA}} + \Delta_{\mathrm{A}} - \Delta_{\mathrm{B}}), min(\gamma_{\mathrm{AB}} + \Delta_{\mathrm{B}} - \Delta_{\mathrm{A}}, R_{TT} + \Delta_{\mathrm{A}}),\\ &min(\gamma_{\mathrm{BA}} + \Delta_{\mathrm{A}} - \Delta_{\mathrm{B}}, R_{TT} + \Delta_{\mathrm{B}})] \end{aligned} \quad (54)$$

Eliminating the terms that are duplicated, we obtain

$$\tau_{upp} = min[max(\gamma_{\mathrm{BA}} + \Delta_{\mathrm{A}} - \Delta_{\mathrm{B}}, \gamma_{\mathrm{AB}} + \Delta_{\mathrm{B}} - \Delta_{\mathrm{A}}), max(R_{TT} + \Delta_{\mathrm{A}}, R_{TT} + \Delta_{\mathrm{B}})] \quad (55)$$

In the same way, an upper bound for the value of τ_c (τ on the correspondent node) is shown in Eq. (56).

$$\tau_{uppC} = max[max(\gamma_{\mathrm{BA}} + \Delta_{\mathrm{A}} - \Delta_{\mathrm{B}}, \gamma_{\mathrm{AB}} + \Delta_{\mathrm{B}} - \Delta_{\mathrm{A}}), max(R_{TT} + \Delta_{\mathrm{A}}, R_{TT} + \Delta_{\mathrm{B}})] \quad (56)$$

Note that the simplifications done to calculate the upper bound and the conditions of the *supremum* in Eqs. (23) and (25) state that $\tau_{max} \leqslant \tau_{upp} \leqslant \tau_{max} + max(\Delta_{\mathrm{A}}, \Delta_{\mathrm{B}})$.

6.2. Upper bound for τ regardless the location of the failure: Unidirectional traffic, Two-Way failure

In a similar way to Section 6.1, we can obtain upper bound Eq. (50), which provided a maximum for τ for Unidirectional traffic, Two-Way failure to obtain an equation that does not depend on the location of the failure.

$$\tau_{upp} < R_{TT} - ((1-\alpha)\gamma_{\mathrm{AB}} + \alpha'\gamma_{\mathrm{BA}}) + \Delta_{\mathrm{A}} \quad (57)$$

On this case, Eq. (57) is maximized by setting $\alpha = 1$ and $\alpha' = 0$. Finally the upper bound value for the τ for this case is presented in Eq. (58).

$$\tau_{upp} < R_{TT} + \Delta_{\mathrm{A}} \quad (58)$$

6.3. Upper bound for the recovery time for bidirectional traffic

We now aim to provide an expression for the Recovery Time regardless the type and location of the failure for Bidirectional traffic. In Section 6.1 we have obtained an expression for τ for Bidirectional traffic, Two-Way failure, that is independent of the location of a failure. We had also obtained two expressions that characterize τ for the case of Bidirectional traffic, One-Way failure (Eqs. (41) and (42)) that already were independent of the location of the failure. The maximum of these expressions correspond to the One-Way failure case. Therefore the upper bound for τ for Bidirectional traffic is

$$\tau_{upp} = max(\gamma_{\mathrm{BA}} + \Delta_{\mathrm{A}} - \Delta_{\mathrm{B}}, \gamma_{\mathrm{AB}} + \Delta_{\mathrm{B}} - \Delta_{\mathrm{A}}) \quad (59)$$

Combining Eqs. (59) and (6), we obtain the upper bound for the Recovery Time:

$$T_{recovery} < T_{RTx} + R_{TT} + max(\gamma_{\mathrm{AB}}, \gamma_{\mathrm{BA}}) + T_{Send} + \tau_{upp} \quad (60)$$

6.4. Upper bound for the recovery time for unidirectional traffic

From Section 6.2, the upper bound of τ for the case of Unidirectional traffic, Two-Way failure corresponds to Eq. (58). The value of τ for Unidirectional traffic, One-Way failure does not depend on the point of failure and is presented in Eq. (52). Then, the upper bound of $T_{recovery}$ for the Unidirectional traffic case is

$$\tau_{upp} = \left\lceil \frac{R_{TT} + T_{KA} - \Delta_{\mathrm{A}}}{\Delta_{\mathrm{A}}} \right\rceil \Delta_{\mathrm{A}} \leqslant R_{TT} + T_{KA} \quad (61)$$

$$T_{recovery} < T_{RTx} + 2R_{TT} + T_{send} + \tau_{upp} \quad (62)$$

7. A case study of the applicability of the results

Eqs. (59)–(62), provide the appropriate values for the timers of REAP to comply with a target Recovery Time given the characteristics of an application and a scenario. We show how the previous results can be applied with a case study: Suppose a bidirectional VoIP (Voice over IP) application for which we require a $T_{recovery}$ value of 2 s, considering that this time is short enough not to make the user think that the call has been disconnected, using a codec which generates a packet each 30 ms. The end-to-end delay available for the communication is upper bounded by 150 ms (symmetrical case), being this value the typical upper bound of the mouth-to-ear delay as specified in [11]. In this scenario, Eq. (59) provides an upper bound for the time required to start the Send Timer (τ) of 150 ms for the first peer detecting the failure. Supposing concurrent path exploration, the time required to recover from a failure corresponds to Eq. (60) and it is equal to $T_{RTx} + 0.3 + T_{Send} + 0.15 + 0.15$ s. The specification of REAP recommends $T_{RTx} = 0.5\ s$, although the only constrain imposed to the T_{RTx} value is that it must be higher than the R_{TT} (Round Trip Time). Now the Send Timer can be set according to the requirements imposed by the application, i.e. assuming the previous $T_{recovery}$ value of 2 s, the Send Timer must be set to 0.9 s. A final check should be performed to be sure that the loss of a small number of packets for reasons such as light congestion does not trigger the exploration process, i.e. check that $\frac{T_{send}}{\Delta_{\mathrm{A}}}$ and $\frac{T_{send}}{\Delta_{\mathrm{B}}}$ are larger than a certain small value such as 3 or 4, so that 3 or 4 data packets should be discarded before the exploration process is started. The main idea behind requiring at least 3 or 4 packets to be lost before entering in the exploration process is to prevent triggering this process as a result of a very small number of uncorrelated events. Or, in other way, this 3 or 4 packets are used to check by sampling that the path has been unavailable during the whole T_{send} period, which is a good criteria to decide that the path should be changed.

Now we analyze a communication with the same characteristics as the example presented above but with a delay of 400 ms. This value is the maximum one-way delay for network planning recommended by [11]. Taking into account the same computations as in the previous example $T_{recovery} < 2.1 + T_{Send}$ s. In this case the previous assumption of a $T_{recovery}$ value of 2 s is impossible to fulfill, being the closest achievable value around 2.4 s, taking for example, $T_{Send} = 10 * \Delta$.

By these simple examples we provide a way for the application to configure the REAP timers according to the desired failure Recovery Time.

8. Generalization for variable rate traffic and TCP

The previous results have been derived on the assumption of constant rate traffic. This section is devoted to analyze the impact of variable rate traffic, and to analyze the specific case of TCP, that can be considered for our purpose as a specific case of constrained variable traffic pattern. The main results of this paper are Eqs. (60) and (62), which present the upper bound for the Recovery Time for Bidirectional and Unidirectional traffic respectively. If we focus on the Unidirectional traffic case (Eq. (62)), the Recovery Time does not depend on the sending inter-packet interval (Δ) of any of the nodes, but only on the characteristics of the link and the configuration of the REAP timers. Hence, it is straightforward that it can be applied to variable rate unidirectional traffic without any additional consideration. However, the Bidirectional traffic case (Eq. (60)) depends on the rates of both nodes. In the case of variable rate traffic, the worst possible condition which yields to the higher Recovery Time occurs when the difference between both traffic rates is the highest possible. In order to obtain the worst possible difference between both traffic rates, it is required to compute $\Delta_{\rm A} - \Delta_{\rm B}$ and $\Delta_{\rm B} - \Delta_{\rm A}$ for the maximum and minimum values of $\Delta_{\rm A}$ and $\Delta_{\rm B}$ respectively, which applied to Eq. (60) results in the upper bound for the Recovery Time.

8.1. TCP

We now discuss how to apply the results for Bidirectional traffic (Section 6.3) to TCP traffic. In the next paragraphs we consider two cases: when only one node is transmitting a large chunk of data, and when both nodes are transmitting a large amount of data.

First, let's consider a TCP application that is transferring data (bulk transfer) from one end host (let's call it A) to the other (let's call it B), i.e., data traffic goes in one direction (A → B) and the return path (B → A) is used only for ACKs. We must consider the worst case for this scenario for Eq. (59) to calculate an upper bound to the Recovery Time. To do so, we analyze the two terms inside the maximum operator separately: $\gamma_{\rm AB} + \Delta_{\rm B} - \Delta_{\rm A}$ and $\gamma_{\rm BA} + \Delta_{\rm A} - \Delta_{\rm B}$. In each term the worst case is given by the values of $\Delta_{\rm A}$ and $\Delta_{\rm B}$ maximizing the difference between them.

The maximum of $\Delta_{\rm B} - \Delta_{\rm A}$ happens when $\Delta_{\rm A}$ is minimum and $\Delta_{\rm B}$ is maximum. The value of $\Delta_{\rm A}$, the inter-packet interval in the sending side, depends in practice on the minimum value allowed by the operating system and MAC access methods, so zero is the worst (minimum) case. $\Delta_{\rm B}$ is the time used by TCP for Delayed ACKs. While TCP ACKs can be sent just when a TCP segment is received, they can be delayed until the Delayed ACK Timer expires, so the value of this timer is the largest value for the inter-packet rate of the node receiving data traffic (see Fig. 16). Therefore, the value of this timer can be used to estimate the worst case for the difference of $\Delta_{\rm B}$ and $\Delta_{\rm A}$:

$$\max\,(\gamma_{\rm AB} + \Delta_{\rm B} - \Delta_{\rm A}) = \gamma_{\rm AB} + T_{delayedACK} \tag{63}$$

This timer is in the order of few hundreds of milliseconds, as recommended in [12] (200 ms in Windows XP[3]).

Now, we calculate the maximum of $\Delta_{\rm A} - \Delta_{\rm B}$. In the worst case $\Delta_{\rm A}$ could be close to the R_{TT} of the communication, if the sending side has sent all the data allowed by its transmission window in negligible time compared to the R_{TT}, and has to wait for an acknowledgement to a previous packet before sending the next packet. Although $\Delta_{\rm B}$ will be usually greater than zero in that case, we can again use zero as a worst case. Therefore:

$$\max(\gamma_{\rm BA} + \Delta_{\rm A} - \Delta_{\rm B}) = \gamma_{\rm BA} + R_{TT} \tag{64}$$

Now, by applying 63 and 64 in (59), we obtain an expression for the worst case of the τ_{upp} for a TCP application that sends bulk traffic from A to B:

$$\tau_{upp} = max(\gamma_{\rm AB} + T_{delayedACK}, \gamma_{\rm BA} + R_{TT}) \tag{65}$$

Using this result in Eq. (60) we get an upper bound for $T_{recovery}$.

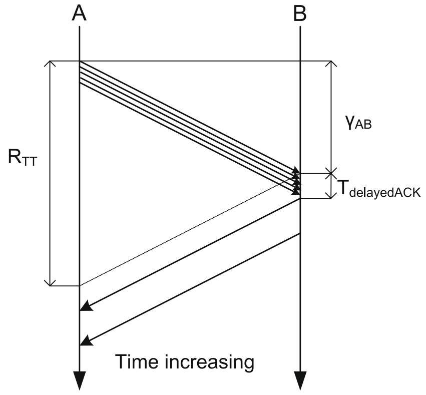


Fig. 16. Exchange of data for TCP application when only one node is transmitting data and the sending site has exhausted its transmission window before receiving an ACK.

The second case considered is a TCP application exchanging bulk traffic in both directions of the communication. Applying the same reasoning as above, we have to consider the two terms in the maximum of Eq. (59). In each of the terms the worst case for the difference between $\Delta_{\rm A}$ and $\Delta_{\rm B}$ and vice versa is given by the Delayed ACK Timer, and occurs when one host is sending packets at an inter-packet interval close to zero and the other end host is waiting the maximum time any of the peers can wait to send a packet. This is a worst case scenario, far from normal operation because the difference between $\Delta_{\rm A}$ and $\Delta_{\rm B}$ will be typically close to zero. Therefore:

$$\tau_{upp} = max(\gamma_{\rm BA} + T_{delayedACK}, \gamma_{\rm AB} + T_{delayedACK}) \tag{66}$$

Then, by introducing (66) in (60), we obtain an expression for the worst case of the $T_{recovery}$ for a TCP application that exchanges bulk bidirectional traffic between A and B.

In conclusion, Eqs. 65, 66, and 60, allow the configuration of REAP timers to achieve objective Recovery Times for worst case scenarios for TCP traffic.

It must be noted that although these results give us the time required by REAP to detect the failure and find a valid path, TCP may need an additional time to start using the new path. This was shown in previous work by the authors [10]. The cause of this mismatch is the congestion control mechanism implemented by TCP. When a failure is detected the time at which the subsequent transmission of segments occurs is driven by the expiration of the Retransmission Timeout (RTO) timer, to reduce the congestion of the network. This timer follows a backoff mechanism, increasing its value when a retransmission occurs. In case REAP detects a failure and finds a suitable path, TCP does not start using the new path immediately, rather it waits until the RTO expires to retransmit the packet.

In [10] the authors proposed a mechanism which, using a cross-layer technique, allowed REAP to reset the RTO of TCP when a new path was found and ready to use. This mechanism has been adopted in a public domain Linux SHIM6 distribution [13]. By the use of this mechanism, the recovery time at TCP level can be modeled as presented in this section.

9. Conclusion

In this paper we have presented an exhaustive analytical study of the time required by REAP to recover from a path failure. We have focused on characterizing the time since the first data packet is lost in any node and the time at which a peer willing to send a packet can do so again, i.e. the $T_{recovery}$ figure of merit. The analysis

[3] http://support.microsoft.com/kb/328890

has considered all the possible situations that may occur for each communication type (bidirectional or unidirectional traffic exchange), different types of failure (One-Way, Two-Way), locations of the failure, transmit start time for each peer, etc. Besides the analysis performed, in which we have proven that some of the expressions provided are optimal upper-bounds (i.e. *supremum* values) for any possible operational scenario, simulations have been used to validate the expressions obtained for several cases. The final result of this process has been two expressions for the upper bound $T_{recovery}$ for bidirectional communication – expression (60) – and unidirectional communication – expression (62) – that are independent of the parameters that cannot be known in advance about the failure that may occur: type and location of the failure. Consequently, these are the most valuable expressions for understanding REAP behavior. These expressions depend on the end-to-end delay of the paths involved, on the values of the timers of REAP (Send Timer and Transmission Timer) and on the inter-packet sending rates of the application. If the traffic pattern generated by an application exchange can be modeled properly, and some assumptions (or real-time measures) about the end-to-end delay can be made, it is easy to configure the REAP timers to assure that communication is restored in a certain time in case of a failure. Such exercise has been presented in Section 7 for a VoIP traffic exchange as a case of study. The use TCP as transport-layer has also been considered, since TCP impose several restrictions to the traffic exchange that can be taking into account by the REAP model. We note that the results obtained are general enough to apply the methodology devised in our work to other failure detection protocols with some similarities to the failure detection module of REAP, such as Bidirectional Forwarding Detection (BFD) [14], Neighbor Unreachability Detection (NUD) [15] or the failure detection mechanism of SCTP [16].

Acknowledgments

The authors would like to thank Jose Felix Kukielka for his helpful contributions to this paper. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement 214994 (CARMEN project). It was also partly funded by the Ministry of Science and Innovation of Spain through Project CONPARTE (TEC2004-05622-C04-03) and project T2C2 (TIN2008-06739-C04-01).

References

[1] E. Nordmark, M. Bagnulo, Shim6: Level 3 Multihoming Shim Protocol for IPv6, IETF RFC 5533, 2009 June.

[2] J. Abley, B. Black, V. Gill, Goals for IPv6 Site-Multihoming Architectures, IETF RFC 3582, 2003 August.

[3] J. Arkko, I. van Beijnum, Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming, IETF RFC 5534, 2009 June.

[4] R. Moskowitz, P. Nikander, Host Identity Protocol (HIP) Architecture, IETF RFC 4423, 2006 May.

[5] T.R. Henderson, Host mobility for IP networks: a comparison, IEEE Networks, 2003 November.

[6] R. Wakikawa, T. Ernst, K. Nagami, Multiple Care-of Addresses Registration, IETF draft; draft-ietf-monami6-multiplecoa-01, October 2006.

[7] T. Kivinen, H. Tschopfening, Design of the IKEv2 Mobility and Multihoming Protocol, IETF RFC 4621, 2006 August.

[8] Marcelo Bagnulo, Alberto Garcia-Martinez, Arturo Azcorra, IPv6 multihoming support in the mobile Internet, IEEE Wireless Communications Magazine 14 (5) (2007) 92–98. October.

[9] S.Barre, O. Bonaventure, Improved path exploration in shim6 based multihoming, in: Proc. ACM SIGCOM Workshop on IPv6 and the Future of the Internet, August 2007.

[10] A. de la Oliva, M. Bagnulo, A. Garcia-Martinez, I. Soto, Performance analysis of the REAchability protocol for IPv6 multihoming, NEW2AN 2007, in: Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking, September 2007.

[11] R. ITU-T, I. Recommend, G. 114, One-way transmission time 18.

[12] R. Braden, Requirements for Internet Hosts-Communication Layers, IETF RFC 1122, October 1989.

[13] Sebastien Barre, Olivier Bonaventure, Implementing SHIM6 using the Linux XFRM framework, in: Routing In Next Generation workshop, Madrid, Spain, 2007.

[14] D. Katz, D. Ward, Bidirectional Forwarding Detection, IETF draft; draft-ietf-bfd-base-07, January 2008.

[15] T. Narten, E. Nordmark, W. Simpson, H. Soliman, Neighbor Discovery for IP Version 6, IETF RFC 4861, September 2007.

[16] R. Stewart, Stream Control Transmission Protocol, IETF RFC 4960, September 2007.