Scientific Research Publishing

Open Journal of Applied Sciences, 2026, 16(11), 4609-4622
https://www.scirp.org/journal/ojapps
ISSN Online: 2165-3925
ISSN Print: 2165-3917

Design of an Adaptive Delta-State CRDT Propagation Mechanism Based on the AIMD Principle

Tanon Lambert Kadjo[1*], Loukou Kouadio Denis[1], Lagasane Ouattara Kra[2], Karamoko Vafle[2], Olivier Asseu[1,3]

[1]INP-HB, Yamoussoukro, Côte d'Ivoire
[2]Université Alassane Ouattara, Bouake, Côte d'Ivoire
[3]ESATIC, Abidjan, Côte d'Ivoire
Email: *tanon.kadjo@inphb.ci

How to cite this paper: Kadjo, T.L., Denis, L.K., Kra, L.O., Vafle, K. and Asseu, O. (2026) Design of an Adaptive Delta-State CRDT Propagation Mechanism Based on the AIMD Principle. *Open Journal of Applied Sciences*, **16**, 4609-4622.
https://doi.org/10.4236/ojapps.2026.1611253

Received: October 20, 2026
Accepted: November 20, 2026
Published: November 23, 2026

Copyright © 2026 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Open Access

Abstract

Real-time collaborative applications rely on the replication of data between multiple nodes, but their synchronization can become costly in constrained network environments. Despite the advantages offered by Delta-State CRDT (δ-CRDT), classical propagation policies rely on static parameters that are not sensitive to variations in bandwidth, latency and packet loss. This article proposes AIMD-δ CRDT, an adaptive mechanism based on the Additive Increase Multiplicative Decrease (AIMD) principle, which dynamically adjusts the grouping threshold of operations from application acknowledgments and timeout expirations. An analytical model of the average threshold, effective bandwidth and synchronization latency is established in steady state. The theoretical analysis shows that the mechanism preserves the convergence properties of the underlying δ-CRDT, remains stable and does not increase the algorithmic complexity order. As the mechanism does not intervene in merger operations, guarantees of convergence are theoretically preserved.

Keywords

CRDT, δ-CRD, AIMD, Adaptive Synchronization, Bandwidth, Distributed Systems

1. Introduction

Real-time collaborative applications rely on data replication between multiple nodes to ensure availability and fault tolerance, while also guaranteeing the con-

vergence of copies modified concurrently by multiple users. Following centralized approaches and Operational Transformation [1], Conflict-Free Replicated Data Types (CRDTs) provided a decentralized solution to this problem by ensuring replica convergence without prior coordination [2]. Among them, Delta-State CRDTs (δ-CRDTs) reduce synchronization costs by propagating only incremental modifications rather than the complete state of replicas [3].

However, this gain remains conditioned by the delta propagation policy, which is generally governed by static parameters (fixed anti-entropy frequency or constant grouping threshold for operations) that are insensitive to actual network conditions. In environments with limited bandwidth, intermittent connectivity, or frequent losses, this rigidity leads to inefficient resource utilization and degrades the performance of collaborative applications.

The Additive Increase Multiplicative Decrease (AIMD) principle, which underpins TCP congestion control [4] and was formalized by [5], offers a proven adaptive control mechanism: a parameter is gradually increased when conditions are favorable, then abruptly reduced as soon as degradation is detected. This principle has already been applied to flow control in anti-entropy protocols for weakly consistent systems [6], but its application to the grouping threshold of δ-CRDT operations remains, to our knowledge, unexplored in the recent literature on CRDT optimization, even though this threshold has a control structure directly comparable to that of a congestion window.

This article bridges this gap by proposing AIMD-δCRDT, a mechanism that applies the AIMD principle to the grouping threshold of δ-CRDT operations, using application acknowledgements and timeout expirations as feedback signals. The plan for implementing our approach in this paper is as follows: i) an adaptive control architecture integrating a dynamic threshold θ into δ-CRDT propagation; ii) a mathematical model relating the loss rate to the average threshold and to bandwidth and latency propagation costs; iii) an algorithmic specification of the mechanism; iv) a theoretical analysis of convergence, threshold stability, and complexity.

The article is organized as follows: Section 2 presents the bibliographic review, previous work; Section 3 details the design and formalization of AIMD-δCRDT, then the conclusion of our article in Section 4.

2. Related Work

The optimization of CRDT synchronization falls under two complementary directions: reducing the amount of information exchanged and adapting the synchronization strategy according to the exchanges performed.

2.1. Reduction of Exchanged Content

A first family of works aims to reduce the volume or redundancy of information exchanged between replicas. The Delta-State CTRDs (δ-CRDTs) of [3] [7] belong to this category; Instead of propagating the full state of a replica, they transmit

only incremental deltas-states, representing the part of the state necessary to account for a local mutation. This delta-mutation significantly reduces the volume of each exchange, especially when the global state becomes large. The associated anti-entropy mechanism makes it possible to propagate these deltas on unreliable channels while maintaining the convergence model of state CRDTs [7].

Enes and colleagues [8] continue this logic by showing that stateful CRDT synchronization algorithms can produce redundant propagations. They use join-decompositions to build more efficient deltas, by transmitting only the elements that are missing in the recipient. ConflictSync of [9] takes a complementary approach: it uses digests to precisely identify the deltas actually needed for synchronization, thus avoiding the transmission of already available data.

These approaches reduce the cost of each sync by optimizing message content. However, they remain insensitive to communication conditions: they do not change the frequency at which synchronizations are triggered. A message, even a compact one, can continue to generate a significant overhead if it is transmitted at a frequency unsuited to the conditions of the network.

2.2. Adapting the Synchronization Strategy

The works of this second family adapt the synchronization policy or mode to the network conditions. In [10], The authors study the synchronization of δ-CDTs in opportunistic networks tolerant to frequent disconnections. This is the case with FlexSync, which is another author who selects, from among several propagation policies, the one appropriate for the application and network context [11]. As for the PS-CRDT of the authors of [12], it favors robust propagation mechanisms against prolonged disconnections in highly volatile environments. This work adapts the synchronization to the network conditions, but in a discrete way (by choice of a policy or mode) rather than by a continuous regulation driven by feedback.

2.3. Research Positioning

The AIMD principle shows, both in congestion control [4] [5] and in its adaptation to anti-entropy protocols [6], that a control parameter can be gradually adjusted based on continuous feedback. None of the works considered here exploits this continuous regulation for the grouping threshold of a δ-CRDT operations: the works aiming to reduce the exchanged content [8] [9] mainly optimize what is transmitted, while the works adapting the synchronization policy ([10] [12]) prioritize the choice of policy or mode. It is at this intersection (δ-CRDT efficiency, continuous sensitivity to network conditions, and closed loop regulation) that our contribution lies, as detailed in Section 3.

3. Design and Formalization of the AIMD-δ CRDT Mechanism

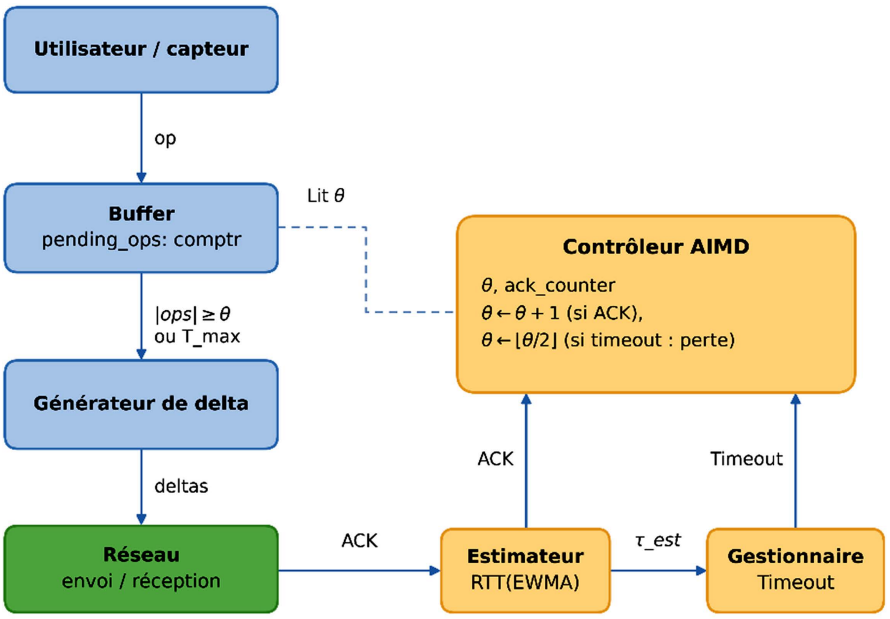
3.1. Architecture and Operation

3.1.1. Architecture

AIMD-δCRDT dynamically adapts the propagation of Delta-State CRDTs accord-

ing to the observed communication conditions. Its architecture (see Figure 1) combines local operations management with an adaptive controller responsible for adjusting the propagation threshold θ and is based on the following components:

- **Operations Buffer**: stores local operations before they are propagated, to allow grouping and reduce the frequency of transmissions.
- **Deltas generator**: builds a delta-state from buffer operations when the threshold θ is reached or when the starvation timer expires.
- **AIMD controller**: the heart of the mechanism, it dynamically adjusts the threshold according to two rules: additive increase ($\theta \leftarrow \theta + \alpha$) when communications are satisfactory, and multiplicative decrease ($\theta \leftarrow \beta \times \theta$, with $0 < \beta < 1$) when a loss is detected.
- **RTT estimator**: continuously calculates the average round-trip time using an Exponentially Weighted Moving Average (EWMA) filter.
- **Timeout manager**: monitors transmitted delta-states; in the absence of an application ACK before the expiration of the period calculated from the estimated RTT, a loss is declared and transmitted to the AIMD controller.
- **Starvation timer**: ensures synchronization even in case of low activity, generating a delta-state if the threshold θ is not reached before the expiration of Tmax.
- **Network interface**: handles the transmission of delta-states and the reception of feedback information (ACK, RTT, timeouts).



- RTT estimé (EWMA) : $\tau_{est} \leftarrow (1-\gamma)\,\tau_{est} + \gamma \cdot RTT_{mesuré}$, avec $\gamma = 0{,}125$
- Timeout : $2 \cdot \tau_{est}$
- Délai maxi d'attente : $T_{max} = 2$ s

Figure 1. Functional architecture of the AIMD-δCRDT mechanism.

This architecture (Figure 1), is organized around two complementary plans: the data plane, responsible for the production, storage, and propagation of updates, and the control plane, responsible for observing communication conditions and dynamically adjusting the propagation threshold. This separation makes it possible to integrate an adaptation mechanism without modifying the internal functioning of the δ-CRDT.

3.1.2. Operation

A locally produced operation is first applied to the CRDT and then placed in the buffer. When the number of operations reaches the threshold θ, a delta is constructed and transmitted. The application-level acknowledgment (ACK) is the main feedback signal. Upon receiving each valid ACK, the controller increments an acknowledgment counter. When this counter reaches the current threshold θ, the grouping threshold is increased additively according to the rule $\theta \leftarrow \theta + \alpha$, and the counter is reset to zero. Conversely, when no ACK is received before the timeout expires, a loss is assumed and the controller applies multiplicative decrease. The unacknowledged delta is retained until its corresponding ACK arrives and is retransmitted if a timeout occurs. The starvation timer prevents low application activity or a high threshold from causing an indefinite synchronization delay. The mechanism thus forms a local control loop that adapts the propagation rate of δ-CRDTs without modifying their operations or merge rules.

Control-design rationale. Reducing the batch size after a timeout may seem counterintuitive, since smaller batches increase the message frequency and therefore the share of header overhead H in the traffic (second term of Eq. (15)). Multiplicative decrease is nevertheless justified by three effects. First, it limits exposure to losses: when a delta of k operations is lost, those k operations are delayed by at least one timeout and their payload $k \cdot s$ must be retransmitted; with a smaller θ, each loss affects fewer operations and the retransmitted volume is smaller. Second, it reduces the waiting delay in the buffer $(\theta - 1)/(2\lambda)$ given by Eq. (17), which partly compensates for the timeout delays accumulated during degraded periods. Third, the reduction is only transient: additive increase restores large batches as soon as acknowledgments arrive regularly, so that header overhead is amortized again when conditions improve. Decrease thus trades a bounded increase in the overhead term for lower latency and a reduced retransmission cost; Eqs. (15) and (19) quantify both sides of this trade-off.

It is, however, appropriate to distinguish the origin of losses. i) Congestion-related losses are correlated with queue buildup and are generally preceded by an increase in the measured RTT. The sending rate is then regulated by the transport layer (for example, TCP congestion control [4]); AIMD-δCRDT is not a congestion controller, and its decrease aims to reduce the volume of each retransmission and the number of operations blocked behind a lost delta, while the floor $\theta \geq 1$ bounds the increase in message rate. ii) Losses due to unreliable connectivity (wireless links, intermittent connections) are random and independent of load. Reducing θ then does not relieve any bottleneck: its only benefit is lower latency

and a reduced retransmitted volume, at the cost of higher header overhead. Under high and persistent random losses, a systematic multiplicative decrease may therefore be too reactive; this behavior was observed in the simulations of [13], where the mechanism consumed more bandwidth than periodic anti-entropy for $p \geq 0.1$, while becoming faster at high loss rates (up to 5.5 times at $p = 0.5$).

3.2. Mathematical Model

The model is built around a single state variable, the grouping threshold θ, which directly influences the frequency of synchronizations and, consequently, the bandwidth consumption and latency. It is based on the AIMD analytical principles introduced by Chiu and Jain [5] and on the probabilistic analysis of congestion control mechanisms developed by Padhye *et al.* [14]. Unlike this work, where AIMD controls the congestion window of a transport protocol, the proposed mechanism applies this principle to the clustering threshold of operations of a δ-CRDT; which leads to an original analytical model.

3.2.1. Modelling Assumptions

- **H1 - Deltas structure**: each delta consists of a set of k serialized operations, whose size grows linearly with k according to $Dk = H + k \times s$, where H is the average fixed message overhead and s the average size of a serialized operation.
- **H2 - Generation of operations**: local operations follow a Poisson process with average rate λ operations per second.
- **H3 - Loss model**: the communication channel is modeled by a Bernoulli channel, each delta being lost with a probability p.
- **H4 - Estimation of the RTT**: the round-trip time is estimated locally by an EWMA filter [4] [15], used to dynamically adjust the delay.
- **H5 - Steady-state**: the analysis is conducted after disappearance of transient effects related to initialization, period during which threshold oscillations become periodic.

These assumptions define the chosen modeling framework; they simplify the analysis while maintaining the essential characteristics of the mechanism.

3.2.2. Evolution of the Adaptive Threshold

In the absence of congestion, the threshold is updated according to the additive increase rule:

$$\theta_{n+1} = \theta_n + \alpha \tag{1}$$

where $\alpha \in \mathrm{N}^*$ is the additive increase step. When a transmission failure is detected by the acknowledgment timeout, the threshold is reduced according to the multiplicative decrease rule:

$$\theta_{n+1} = \beta\theta_n,\ 0 < \beta < 1 \tag{2}$$

The alternation of these two rules produces a sawtooth evolution characteristic of AIMD algorithms. At the start of each cycle, the threshold is $\theta_{\min} = \beta \cdot \theta_{\max}$, where $\theta_{\max}$ represents the maximum value reached immediately before congestion was

detected.

$$\theta_{\min} = \beta\theta_{\max} \tag{3}$$

3.2.3. Number of Transmissions per Cycle and Maximum Threshold

During the additive increase phase, the threshold increases by one unit with each synchronization. The total number of transmissions during a cycle corresponds to the sum of the successive values of the threshold between $\theta_{\min}$ et $\theta_{\max}$:

$$N_{\text{cycle}} = \sum_{i=\theta_{\min}}^{\theta_{\max}} i. \tag{4}$$

Using the formula for the sum of consecutive integers and relation (3), we obtain:

$$N_{\text{cycle}} = \frac{1-\beta^2}{2}\theta_{\max}^2 + \frac{1+\beta}{2}\theta_{\max} \tag{5}$$

For large values of the threshold ($\theta_{\max} \gg 1$, steady state, $0 < \beta < 1$), the ratio between the linear term and the quadratic term of (5) tends to 0, which allows the asymptotic approximation:

$$N_{\text{cycle}} \approx \frac{1-\beta^2}{2}\theta_{\max}^2. \tag{6}$$

In accordance with the H3 hypothesis, losses are modeled by a Bernoulli law for the parameter p; the average number of transmissions before the first congestion follows a geometric law of expectation $1/p$ [14], that is, $N_{\text{cycle}} \approx 1/p$. When combining this relationship with (6), it comes to:

$$\theta_{\max} = \sqrt{\frac{2}{\left(1-\beta^2\right)p}} \tag{7}$$

For $\beta = 0.5$ (the value used in this study), this expression becomes:

$$\boxed{\theta_{\max} = \sqrt{\frac{8}{3p}}} \tag{8}$$

This expression shows that the maximum value of the threshold increases when the probability of loss decreases, allowing a greater grouping of operations before their propagation.

3.2.4. Mean Steady-State Threshold

In steady state, θ is treated as a random variable of expectation $\mathbb{E}[\theta]$. During the additive increase phase, the threshold evolves linearly between $\theta_{\min}$ and $\theta_{\max}$, which makes it possible to approach the average threshold by the arithmetic mean of these two bounds:

$$\mathbb{E}[\theta] \approx \frac{\theta_{\min} + \theta_{\max}}{2}. \tag{9}$$

Using the relationship (3) and then the expression of $\theta_{\max}$ given by (7), we obtain successively:

$$\mathbb{E}[\theta] = \frac{1+\beta}{2}\theta_{\max} = \frac{1+\beta}{2}\sqrt{\frac{2}{(1-\beta^2)p}} \tag{10}$$

For $\beta = 0.5$, this expression is simplified to:

$$\boxed{\mathbb{E}[\theta] \approx \sqrt{\frac{3}{2p}}} \tag{11}$$

This relationship constitutes the central result of the analytical model: it establishes an explicit link between the average clustering threshold and the probability of loss p.

3.2.5. Effective Bandwidth

The consumed bandwidth is broken down into two components: the useful data (B_{data}), corresponding to the transported operations, and the average fixed overhead (B_{ov}) associated with each synchronization message:

$$B_{\text{eff}} = B_{\text{data}} + B_{\text{ov}} \tag{12}$$

Under the H2 hypothesis, each operation has an average serialized size s; in the presence of a loss probability p, each message requires on average $\frac{1}{1-p}$ transmission attempts, hence:

$$B_{\text{data}} = \frac{\lambda s}{1-p} \tag{13}$$

The average rate of messages sent is obtained by dividing the average transaction rate λ by the average number of transactions grouped, that is to say; the contribution of the communication overhead becomes: $\mathbb{E}[\theta]$

$$B_{\text{ov}} = \frac{\lambda H}{\mathbb{E}[\theta](1-p)} \tag{14}$$

By replacing $\mathbb{E}[\theta]$ with the expression (11), we obtain the complete analytical model of the effective bandwidth:

$$\boxed{B_{\text{eff}} = \frac{\lambda s}{1-p} + \frac{\lambda H}{1-p}\sqrt{\frac{2p}{3}}} \tag{15}$$

This expression highlights two components: the first, related to the volume of application data, is independent of the mechanism; the second, related to the frequency of synchronizations, is directly regulated by the AIMD controller. When p decreases, the average threshold increases, which amortizes the fixed cost of headers over a larger number of operations; conversely, when p increases, the threshold automatically decreases, which increases the frequency of synchronizations to maintain a more reactive propagation.

Proposition 1. *At constant application load, the bandwidth consumption of the AIMD-δ CRDT mechanism is mainly governed by the frequency of synchronizations, which directly depends on the adaptive clustering threshold. The mechanism thus optimizes communications without modifying data structures or δ-*

CRDT merging operations.

3.2.6. Synchronization Latency

Synchronization latency results from two components: the waiting time in the buffer (L_{buffer}), necessary to accumulate operations, and the transmission time on the network (L_{network}):

$$L_{\text{total}} = L_{\text{buffer}} + L_{\text{network}} \tag{16}$$

Under the hypothesis H2 (Poisson process), and using the property that the arrival times of θ operations over a given interval are distributed like the order statistics of independent uniform variables [16], we consider a synchronization cycle where T designates the time between the arrival of the first operation and the emission of the corresponding message: we have $\mathbb{E}[T] = \frac{\theta - 1}{\lambda}$, and the average arrival time of the operations is equal to $T/2$, the average time spent by an operation in the buffer is:

$$L_{\text{buffer}} = \frac{\mathbb{E}[T]}{2} = \frac{\theta - 1}{2\lambda} \tag{17}$$

A message not acknowledged before the expiration of the timeout period is retransmitted; the average number of attempts follows a geometric distribution of parameters $(1-p)$, of expectation $1/(1-p)$, hence:

$$L_{\text{network}} = \frac{\tau_{\text{est}}}{1-p} \tag{18}$$

where τ_{est} refers to the round-trip time estimated by the EWMA filter [15]. By combining (17) et (18), we obtain the analytical model of the average synchronization latency:

$$L_{\text{total}} = \frac{\sqrt{\frac{3}{2p}} - 1}{2\lambda} + \frac{\tau_{\text{est}}}{1-p} \tag{19}$$

This relationship highlights the compromise introduced by the mechanism: when p is low, the threshold increases and messages are generated less frequently, which reduces communication overhead but slightly increases waiting time in the buffer; when p increases, the threshold decreases, which reduces the wait in the buffer but increases the frequency of retransmissions.

Proposition 2. *The average synchronization latency of the AIMD-δ CRDT mechanism results from the trade-off between the accumulation time of operations in the buffer and the transmission time on the network. By dynamically adapting the grouping threshold, the AIMD controller automatically adjusts this trade-off according to the observed communication conditions*.

3.3. Mechanism Algorithm

The following algorithm describes the behavior of a replica executing AIMD-δ CRDT. It is based on an event-driven logic where controller decisions are triggered by local operations, received acknowledgements and timing events; ACKs

and timeouts serve as feedback signals to adapt dynamically θ.

Inputs: α (additive increase step, $\alpha = 1$); β (multiplicative decrease factor, $\beta = 0.5$); θ_0 (initial threshold); Tmax (maximum waiting time); Rmax (maximum retransmission attempts).

Global variables: $\theta \leftarrow \theta_0$; Buffer $\leftarrow \varnothing$; Pending $\leftarrow \varnothing$; RTTest $\leftarrow$ RTT_initial; seq $\leftarrow$ 0; ack_counter $\leftarrow$ 0.

Event E1 – onLocalOperation (op): *apply the local mutation to the CRDT*; *insert op into the Buffer*; *if Size* (*Buffer*) $\geq \theta$, *call SEND_DELTA* ().

Event E2 – onAck (delta_id, measuredRTT): update $RTTest \leftarrow (1-\gamma)\cdot RTTest + \gamma \cdot measuredRTT$; *if Pending*[*delta_id*] *exists*, *Pending.remove*[*id*]; *cancel its timeout*; *ack_counter* $\leftarrow$ *ack_counter* + 1; *if ack_counter* $\geq \theta$, $\theta \leftarrow \theta + \alpha$ *and ack_counter* $\leftarrow$ 0.

Event E3 - onTimeout(delta_id): *if Pending*[*delta_id*] *exists*, *then* $\theta \leftarrow max(1, \beta \times \theta)$; *ack_counter* $\leftarrow$ 0; *if Pending*[*delta_id*].*retry_count* < *R_max*, *increment its retry_count*, *retransmit* (*delta_id*, *Pending*[*delta_id*].*delta*), *and restart its timer using RTT_est*; *otherwise*, *trigger the recovery mechanism*.

Event E4 - onFamine(): *if Buffer* $\neq \varnothing$, *call SEND_DELTA* ().

Procedure SEND_DELTA (): *if Buffer* $\neq \varnothing$, *seq* $\leftarrow$ *seq* + 1; *id* $\leftarrow$ *seq*; $\Delta \leftarrow$ *build_delta*(*Buffer*); *store Pending*[*id*] $\leftarrow \Delta$; *send* (*id*, Δ); *Buffer* $\leftarrow \varnothing$; *start_timeout*(*id*).

This algorithm adjusts the propagation threshold using synchronization feedback without modifying the operations or merge rules of the underlying CRDT.

3.4. Multi-Node Topology and Per-Peer Adaptation

In a multi-node topology, a replica may synchronize simultaneously with several peers experiencing different network conditions. Using a single global threshold could prevent the mechanism from adapting appropriately to each link. Therefore, AIMD-δCRDT maintains an independent grouping threshold θ_{ij} for each directed link from replica R_i to replica R_j.

Each link maintains its own acknowledgment counter, RTT estimate, timeout state, and pending-delta table. When an acknowledgment (ACK) is received from a given peer, only the corresponding controller state is updated. If the acknowledgment counter reaches the current threshold θ_{ij}, that threshold is increased according to the additive rule. Conversely, a timeout triggers multiplicative decrease for the affected link only, and the corresponding unacknowledged delta is retransmitted. This separation prevents network conditions on one link from directly modifying the propagation threshold used for another peer.

The multi-node behavior of AIMD-δCRDT has been partially investigated in our previous simulation study [13], which considered a four-replica K4 topology with adaptive thresholds maintained per link. This study provides a basis for examining the mechanism beyond a single communication link. However, its results do not establish the behavior of the mechanism under all heterogeneous or time-varying network conditions, which remain to be investigated in further experiments.

3.5. Theoretical Analysis of the Mechanism Properties

3.5.1. Preserving Strong Eventual Consistency

Proposition 3. Under the assumption that the necessary deltas are eventually delivered, the introduction of the AIMD-δ CRDT mechanism does not alter the Strong Eventual Consistency (SEC) property of the underlying δ-CRDT.

Justification: the mechanism acts only on the deltas propagation policy—it modifies the moment at which they are transmitted, without intervening on their content, the elementary operations of the CRDT, the merging function, or the causal relationships between updates. The properties of the merge function that ensure the convergence of δ-CRDT [2] [7] therefore remain unchanged. Thus, if the possible delivery conditions required by the δ-CRDT are met, any replica eventually receiving the same set of deltas converges to the same end state. This analysis establishes the absence of controller interference with the convergence mechanism; it does not constitute an experimental validation of inter-replica convergence.

3.5.2. Controller Stability

Proposition 4. *Under the assumptions H*1 *to H*5, *the AIMD-*δ *CRDT controller exhibits stable behavior and a bounded adaptive threshold.*

Justification: the alternation between additive increase and multiplicative decrease produces successive cycles characteristic of the AIMD mechanism. According to equation (10), the mean value of the threshold remains finite for any loss probability $p > 0$: the controller oscillates around a mean value without divergence, in accordance with the theoretical behavior of AIMD mechanisms [5] [14].

3.5.3. Simulation-Based Evaluation under Time-Varying Network Conditions

To complement the analytical model, two scenarios were simulated over ten independent runs: a temporary increase in packet loss from 1% to 20%, and an RTT spike from 50 ms to 250 ms. The results are summarized in **Table 1**.

Table 1. Simulation results under time-varying network conditions.

Metric	Variable loss	RTT spike
Mean threshold (\theta)	9.24	6.85
Bandwidth (bytes/s)	1801.60	1751.31
Mean retransmissions	105.5	43.5
Mean ACK latency (ms)	49.98	84.35
95th-percentile latency (ms)	66.40	262.68
Abandoned deltas	0	0
Pending operations at the end	0	1.1

Under variable packet loss, the threshold decreased during the disturbance

and increased after network conditions improved. The mean ACK latency remained close to 50 ms, although retransmissions increased. Under the RTT spike, mean ACK latency reached 84.35 ms, with a 95th-percentile latency of 262.68 ms. An average of 1.1 operations remained pending at the end of the evaluation window.

These results characterize the controller's behavior under the two simulated disturbances but do not establish its stability or recovery rate. Such conclusions require further analysis of threshold time series across independent runs.

These experiments complement the previously published study [13], which compared AIMD-driven propagation with fixed-threshold and periodic anti-entropy policies in a four-replica mesh topology and reported a 25% - 36% bandwidth reduction relative to the fixed-threshold policy. While the published study evaluated performance under its tested configurations, the present experiments focus on time-varying network conditions. Further work is needed to assess convergence under dynamic conditions, heterogeneous links, intermittent connectivity, and real-world network traces.

3.5.4. Algorithmic Complexity

The analysis focuses on the main operations carried out by a replica. The processing of a local operation (buffer insertion, threshold comparison) and the processing of an ACK (EWMA update, threshold comparison) are limited to elementary operations in O(1), as is the management of a timeout (multiplication and allocation). The construction of a delta requires running and serializing the k operations of the buffer, ie a complexity O(k), where $k \approx \mathrm{E}[\theta]$ in steady state; a bounded value according to Proposition 4. The spatial complexity is also O(k), with the buffer being the main cost. The complexity analysis of AIMD-δCRDT is summarized in Table 2.

Table 2. Algorithmic complexity of the AIMD-δCRDT mechanism.

Operation	Complexity
Processing a local operation	O(1)
Processing of an acknowledgment (ACK)	O(1)
Managing a timeout	O(1)
Construction of a delta	O(k)
Spatial complexity	O(k)

The addition of the AIMD controller therefore does not change the asymptotic complexity of synchronization: control operations remain in O(1) and the construction of a delta remains in O(k), already necessary for the propagation of δ-CRDTs. The mechanism thus exchanges a small local overhead for a reduction in the number of transmissions, without increasing the order of complexity.

4. Conclusions and Perspectives

This paper proposed AIMD-δ CRDT, a mechanism that applies the AIMD principle to the grouping threshold of δ-CRDT operations, using application acknowledgments and timeout expirations as feedback signals. It acts only on the propagation policy, without modifying the operations or merging rules of the δ-CRDT. Our contributions are: i) an architecture separating the data plane and the control plane; ii) an analytical model linking the loss rate to the average threshold, bandwidth, and latency; iii) an event-based algorithmic specification; iv) an establishing analysis. The model highlights the trade-off between message overhead damping and buffer accumulation delay. These results are valid approximations under H1 - H6 (notably $\theta_{\max} \gg 1$). Additional simulations involving variable packet loss and temporary RTT increases provide preliminary observations of the controller's behavior under changing network conditions. However, they do not establish practical stability, the absence of irregular oscillations, or convergence among multiple replicas under dynamic conditions.

The perspectives include correlated losses, dynamic adaptation of α and β, multi-replica extension, and evaluation on real network traces.

Author Contributions

Tanon Lambert Kadjo: Conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing, visualization, funding acquisition. **Loukou Kouadio Denis**: Conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing, visualization, supervision, project administration, funding acquisition. **Lagasane Ouattara Kra**: investigation, resources, data curation, writing—original draft preparation, writing—review and editing, funding acquisition. **Karamoko Vafle**: Conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing, visualiza-tion, supervision, project administration, funding acquisition. **Asseu**: **Olivier Pascal Kouame:** Validation, formal analysis, writing—original draft preparation, writing—review and editing, visualiza-tion, supervision, project administration.

Conflicts of Interest

The authors declare that they have no conflict of interest regarding the publication of this article.

References

[1] Ellis, C.A. and Gibbs, S.J. (1989). Concurrency Control in Groupware Systems. *Proceedings of the* 1989 *ACM SIGMOD International Conference on Management of Data*, Portland, 31 May-2 June 1989, 399-407. https://doi.org/10.1145/67544.66963

[2] Shapiro, M., Preguiça, N., Baquero, C. and Zawirski, M. (2011) Conflict-Free Repli-

cated Data Types. In: Défago, X., *et al*., Eds., *Stabilization, Safety, and Security of Distributed Systems*, Springer, 386-400. https://doi.org/10.1007/978-3-642-24550-3_29

[3] Almeida, P.S., Shoker, A. and Baquero, C. (2015) Efficient State-Based CRDTs by Delta-Mutation. In: Bouajjani, A. and Fauconnier, H., Eds., *Networked Systems: Third International Conference*, Springer International Publishing, 62-76. https://doi.org/10.1007/978-3-319-26850-7_5

[4] Jacobson, V. (1988) Congestion Avoidance and Control. *ACM SIGCOMM Computer Communication Review*, **18**, 314-329. https://doi.org/10.1145/52325.52356

[5] Chiu, D. and Jain, R. (1989) Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks and ISDN Systems*, **17**, 1-14. https://doi.org/10.1016/0169-7552(89)90019-6

[6] van Renesse, R., Dumitriu, D., Gough, V. and Thomas, C. (2008) Efficient Reconciliation and Flow Control for Anti-Entropy Protocols. *Proceedings of the* 2*nd Workshop on Large-Scale Distributed Systems and Middleware*, New York, 15-17 September 2008, 6:1-6:7. https://doi.org/10.1145/1529974.1529983

[7] Almeida, P.S., Shoker, A. and Baquero, C. (2018) Delta State Replicated Data Types. *Journal of Parallel and Distributed Computing*, **111**, 162-173. https://doi.org/10.1016/j.jpdc.2017.08.003

[8] Enes, V., Almeida, P.S., Baquero, C. and Leitao, J. (2019) Efficient Synchronization of State-Based CRDTs. 2019 *IEEE* 35*th International Conference on Data Engineering* (*ICDE*), 8-11 April 2019, 148-159. https://doi.org/10.1109/icde.2019.00022

[9] Gomes, P.S., Rodrigues, M.B. and Baquero, C. (2025) ConflictSync: Bandwidth Efficient Synchronization of Divergent State. arXiv: 2505.01144.

[10] Guidec, F., Mahéo, Y. and Noûs, C. (2023) Supporting Conflict-Free Replicated Data Types in Opportunistic Networks. *Peer-to-Peer Networking and Applications*, **16**, 395-419. https://doi.org/10.1007/s12083-022-01404-6

[11] Xu, Z., Dong, Y., Lou, J., Wang, Y. and Fu, Y. (2024) Edge AI-Driven Neural Network Predictions for Replica Sync Optimization. *Applied Soft Computing*, **165**, Article ID: 112083. https://doi.org/10.1016/j.asoc.2024.112083

[12] Barreto, A., Paulino, H., Silva, J.A. and Preguiça, N. (2023) PS-CRDTs: CRDTs in Highly Volatile Environments. *Future Generation Computer Systems*, **141**, 755-767. https://doi.org/10.1016/j.future.2022.12.013

[13] Denis, L.K., Kra, L.O., Vafle, K. and Asseu, O. (2026) Aimd-Driven Delta-CRDT Propagation: A Simulation of the Trade-Off between Network Reactivity and Load. https://ejaet.com/PDF/13-10/EJAET-13-10-22-29.pdf

[14] Padhye, J., Firoiu, V., Towsley, D.F. and Kurose, J.F. (2000) Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation. *IEEE/ACM Transactions on Networking*, **8**, 133-145. https://doi.org/10.1109/90.842137

[15] Floyd, S. and Jacobson, V. (1993) Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, **1**, 397-413. https://doi.org/10.1109/90.251892

[16] Ross, S.M. (2014) Introduction to Probability Models. 11th Edition, Academic Press. https://doi.org/10.1016/b978-0-12-407948-9.00001-3