VFC for Cooperative Work

(extended abstract of the MSc dissertation)

João Costa
Departamento de Engenharia Informática
Instituto Superior Técnico

Advisor: Professor Luís Veiga
Co-Advisor: Professor Paulo Ferreira

Abstract—**Although cooperative work is widely used in a daily basis, nonetheless cooperative work tools are not being used as their potential would suggest. Even though the reasons behind this fact are not completely clear, the truth is that users, sometimes, see current cooperative work tools as a burden to their working activities.**

To work around this problem, we exploit two concepts: the notion of locality-awareness and the notion of continuous consistency model. The first refers to the ability of a technique to take in consideration the location of users in the making of decisions. The second corresponds to a notion of a mid-term solution between strong and weak consistency, which controls the divergence limits to find the most adequate balancing between availability and consistency. These two notions, when combined enable a powerful technique which bounds divergence limits on a per-user basis depending on the location in the replicated objects. This is what the Vector-Field Consistency (VFC) model does. However this model was designed for the environment of distributed ad-hoc gaming. In this work we propose VFC for Cooperative Work, an adaptation of the original VFC model to the environment of cooperative work, namely document-based cooperative work.

In this work, we present VFC for Cooperative Work, an adaptation of the VFC original model. We also define how a VFC powered system which is capable of enforcing different divergence limits for each user depending on their location in the replicated object. Later, the VFC for Cooperative Work model was applied to a Latex editor, and the resulting prototype was used to retrieve results to define the success of this work and the merits and flaws of this model.

I. Introduction

In everyday life, it is more and more common to perform *cooperative*[1] *work* to carry out some task. For example, the cooperative production of documents (e.g. articles, presentations, financial reports) is a daily chore in enterprises [1]. Also, wikis, the massive cooperative editor by excellence, are between the most used platforms on the Internet.

If writing is always a long and complex process, *cooperative writing* is an even more complex and difficult process. Cooperatively writing a text can indeed shorten the duration of the writing task if teams work well and so members do not hinder each others work. However, if on the contrary teams do not function well by themselves, the additional effort needed to make them work may not pay off.

On contrary to what may be general belief, *cooperative work tools* are not as used as much their potential suggests [1]. Causing this, may be the general impression that these tools represent more of a burden than a relief. Although this is true in some cases, the major obstacle is still the reluctance of experienced users in changing their everyday tools and methods of collaboration.

Nevertheless, cooperative tools are little by little becoming more accepted and users are starting to see their potential. However, the *replication schemes* behind popular tools are still very conservative. Following the example of Wikipedia, it is implemented above a roughly centralized infrastructure and with a very restrictive conflict resolver mechanism. More, supporting such infrastructures can have huge costs when scaling these tools to a great number of users [2], [3], [4], [5].

Nowadays, popular tools do not try to do an intelligent management of updates. Instead, these tools use mostly all-or-nothing approaches. However, they could try to reason about the importance of each update and perform a selective scheduling based on this importance. This way, the user experience would be improved, since he would not be hindered by frequent but uninteresting updates. Also, since the selective scheduling would result in the delay of less important updates, savings in used network resources could be accomplished by merging of overlapping updates.

This work proposes the adaptation of a consistency model more coherent with the cooperative tools paradigm. First, it proposes the enforcement of a *continuous consistency model* to better meet the requirements of these tools. Also, we discuss how, by combining the notion of *locality-awareness* with the continuous consistency model, we are able to adapt the consistency requirements to the current edition location of each user, within the document, and other points of registered interest. To implement these concepts, the *Vector-Field Consistency* [6] (VFC) algorithm was adapted and implemented to address the desired context.

Since VFC was designed to the gaming environment, it was necessary to port it to the world of document-based cooperative work, namely translate the concepts of location

[1]Although some authors suggest that there is in fact a difference in this field between the terms *cooperation* and *collaboration*, in this work there will be no distinction.

and distance between users' locations, which are pretty straightforward in its original form. Following, *location* is defined as *the place in the document semantics where the user is editing* and *distance* as *the distance between the semantic regions being edited and other points of interest*.

Next, in section II, we overview the consistency maintenance in distributed systems. In addition, we see in more detail several continuous consistency models. Finally we look at the concept of Operation Commutativity. Then, in section III, we detail the architecture that describes the solution. Following, in section IV, a series of quantitative, qualitative and comparative evaluation parameters are presented and their results analyzed. Finally, the section V concludes this work.

II. Related Work

The balance between consistency and availability is a characteristic that separates systems in two opposite classes: *optimistic replication systems* [7] (section II-B) and *pessimistic replication systems* (section II-A). However, sometimes neither the pessimistic nor the unbounded optimistic approaches are acceptable to applications. Thus, it may be beneficial to explore the semantic space between the two alternatives. Hence, in section II-C, we present the existing *continuous consistency models*.

A. Pessimistic Approach

Pessimistic replication systems provide single copy consistency by allowing only one user at a time to perform alterations to a replica. Although these systems offer guaranties of strong consistency between replicas, when ported to a wide-area network such as the Internet, these algorithms cannot provide good performance and availability. Also, they are unable to provide the freedom of edition desired for this work.

B. Optimistic Approach

Optimistic Replication assumes that conflicts will be extremely rare and can be fixed later whenever they appear. For this reason, optimistic algorithms do not require *a priori synchronization* with the other replicas to perform an update. In result, these systems offer greater *availability*, *flexibility*, *scale better* and enable *asynchronous collaboration* even in wide-area environments. Replicas may only converge eventually and so, these algorithms can only be deployed in systems that can support partially inconsistent data, even if only temporarily.

C. Continuous Consistency Models

Designers of replicated systems conventionally choose between strong and optimistic consistency models. Although sometimes, neither the performance overheads imposed by strong consistency neither the lack of limits for inconsistency are acceptable to applications. In such cases, it is appropriate to explore the semantic space between these two alternatives. The fundamental idea behind these *continuous consistency models* is that this space is a continuum parametrized by the *distance* between replicas. This distance is zero for strong consistency and infinite for optimistic consistency. The distance measure can be used to provide a per-replica consistency based on the expected amount of conflicting updates.

Leverage of the consistency space continuum allows systems to correctly balance applications availability and consistency. This balance is affected by factors like application workload, read/write ratios, probability of simultaneous writes, network latency, bandwidth, error rates, etc. Now, we present [8], [6], [9] three different models with great expressive power, that follow this approach.

1) The TACT framework: In [8] is presented a middleware called TACT, which enables applications to quantify their consistency requirements. With TACT, applications need to specify their *conits*, i.e. the physical or logical unit of consistency. Then, the quantification of divergence boundaries is made on a per-replica basis through the use of three metrics, *Numerical Error*, *Order Error*, and *Staleness*. The first metric, *Numerical Error* bounds the difference between the local value of the conit and the value of the 'final image'. *Order Error* bounds the maximum number of tentative writes at a replica. Finally, *Staleness* is the maximum value of time between the last seen local write and the current time.

This model enables the definition of per-replica consistency bounds, which allows systems to greatly adapt to their consistency requirements. For instance, one replica with limited network access may relax its consistency limits. Oppositely, in a replica with faster links it may be viable to impose a stronger consistency. This way, applications can have significant performance improvements without compromising correction.

2) The Vector-Field Consistency model: Vector Field Consistency (*VFC*) [6] is a consistency model that enables replicas to define their consistency requirements in a continuous consistency spectrum. Other than simply *bounding divergence* on a per-replica basis, VFC allows more powerful consistency enforcement policies. For example, using VFC, the maximum allowed difference between two replicas can be dynamically changed during the execution of an application.

Indeed, the novelty of the VFC model is that it combines divergence bounding with the notion of *locality-awareness* to improve the availability and user experience while effectively reducing bandwidth usage. To understand how locality-awareness affects this model, the concept of *pivot* must be introduced. Pivots roughly correspond to each user's observation points within the data. Also, consistency between replicas should strengthen as the distance between their pivots shortens. To define these mutable divergence bounds, around pivots there are several concentric ring-shaped *consistency zones* with increasing distance (radius) and decreasing consistency requirements (increasing divergence bounds). Then, in each zone, like in the TACT framework, programmers use a 3-dimensional vector: *time*, *sequence*, *value*. These boundaries should be specified in a way that does not compromise the user's experience.

The VFC model is extremely flexible, which allows it to be used in a wide variety of systems with very different consistency enforcement policies. Also, VFC effectively reduces the network bandwidth requirements by selectively choosing which updates are more important to which replicas and delaying less important ones, possibly omitting some superseded by later updates.

In comparison with VFC, the TACT framework can enforce the same consistency scenarios but, since it does not support locality-awareness, it cannot be applied to scenarios where consistency depends on the notion of each user position within the data.

3) Data aware Connectivity: The interest about the concept of *data-aware connectivity* system [9] is that it is used to determine the *quality* of a replicated object. Then, unlike in the previously presented works, using the ascertained current value of quality, the system *regulates connectivity* to enforce the divergence/quality bounds. This technique can be very useful in environments with great constraints like mobile environments.

Quality is influenced, among other criteria by its *freshness*, *consistency* and possibility of *rapid commitment*. An important reason for being up to the system to determine replica quality is the fact that these metrics, although intuitive to the attentive user, are not easy to evaluate by the human user. Thus, it is less error-prone to let the system deal with connectivity issues and provide the user an indicative value for the quality of replica so he can decide if he wants to perform a certain task.

D. Operation Commutativity

In the cooperative editing environment, it is natural that replicas diverge if they do not execute operations in the same order. To address the problem of replica converge there are several techniques. One of them is the *Operational Transformation* but, as said in [10], OT is too "*complex and error-prone*". An alternative solution, and the one in focus in this section, is *Operation Commutativity*, the condition that every pair of operations is in commutative relation.

Operation Commutativity aims to the automatic convergence of replicas, i.e. convergence without the need of any complex concurrency control (e.g. lock or serialization). To achieve automatic convergence, it is sufficient the use of a *Commutative Replicated Data Type* (CRDT), as it was baptised [11].

The CRDT approach considers that a document is formed as a sequence of atoms each univocally described by an identifier that does never change during the life span of a document. An atom can be any non editable element like a character or a graphics file. The space of identifiers must be dense and their total order must reflect the order of appearance of atoms in the document. These requirements suggest that rational or real numbers could be used as identifiers, however, they would require infinite precision which is not viable.

The *TreeDoc* [11], [10] is an implementation of the structure of identifiers based on binary trees that represent the document elements/atoms. The total order of those elements can be translated from walking the tree in infix order, which means that the identifiers can be obtained from the tree paths.

III. Solution

In a *cooperative system* which applies a *strict or total consistency model*, as new updates are issued by clients, they are immediately sent to the server. This approach works well for systems with a reduced number of clients and a small rate of creation of new updates. But as the rate at which updates are generated and the number of clients escalate, servers in these systems become a serious bottleneck to the overall system performance since they get overloaded with requests and the time required to respond to them increases.

The solution to this bottleneck problem is based on the decision of which updates must be immediately sent and which can be stored for a while and only later be sent. If we can make this decision, we will be able to combine the temporarily stored updates into lesser and overall smaller ones. Consequentially, we will reduce the usage of network resources and prevent the server form becoming flooded with requests which it cannot respond to in an acceptable time.

A. VFC for Cooperative Work

The great interest about this adaptation is to see how we can attempt to infer which contents are more relevant to the user, or, in this case, the editor, based on its position in the *structure of the distributed contents* (for example, in the *chapter* V, *section* Integration, *subsection* Implementation Issues).

To adapt the VFC model, which was initially created for a totally different environment, its main concepts must be adjusted, namely *replicated objects*, *pivots*, *consistency zones* and *distance*. Also it is not entirely straightforward how to determine to which objects, *consistency vectors* will be applied.

In the scene of cooperative edition of documents, the state of the world, or in other words, the state of the document will be given by the sum of the states of every of its divisions. These divisions can go from entire regions (chapters, sections, paragraphs, etc.) to simple lines or even characters. For architectural reasons we defined such objects (replicated units) as single characters.

Pivots roughly correspond to each user's observation point. We define the pivot position by its place in the structure of the document.

By having a different set of pivots for each user, changes in the document will affect each user differently. This variance is defined by the *distance* between user pivots and the various replicated objects. In more practical terms, consistency between the contents in a semantic region of the document should weaken as the distance to the user pivot grows. Thus, distance is defined as the distance between the semantic regions being edited and the closest pivots.

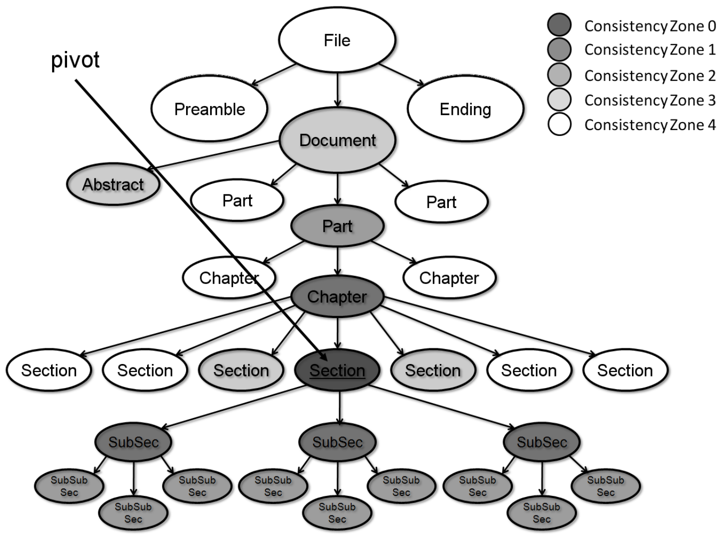


Figure 1. Consistency Zones around a Section

To enforce the different consistency levels, around the pivots of each user we define several *consistency zones*. Consistency zones represent zones in the document in which replicated objects are at a well defined distance to the user pivots and have a certain level of importance to the user's own editions. Thus, consistency zones must be defined according to the document structure. Hence, we can determine which structural divisions of the document are closer to each other and which are farther. In Figure 1 we can see how consistency zones can be defined according to the document structure.

A defining difference between the original VFC model and its adaptation to the cooperative work scenario is that, in the later, the boundaries of the replicated contents change. Meaning that in the gaming scenario the game map coordinates were immutable. But in the cooperative work scenario, updates can change the very structure of the document which is the basis for the locality-awareness in the VFC model.

In the VFC model, for each consistency zone, we define a *consistency vector* to enforce *divergency limits* for objects inside the zone. The 3-dimensional consistency vector [θ, σ, ν] bounds the maximum divergency for a replicated object respectively in terms of *time* (θ, maximum time without updates), *sequence* (σ, maximum number of unseen updates) and *value* (ν, maximum difference between the user and most up-to-date versions). To be in accordance with to the objectives of this adaptation, this limits have to be enforced on a per-region basis.

In this work we defined five different consistency zones (distances from 0 to 4). For each one, we defined a specific consistency vector with the time, sequence and value limits (Table I). Each region belonging to these consistency zones is related to the pivot region in a different way. We now give an intuitive description of those relations that will help define consistency zones in a similar type of structure:

Zone	*Time* (θ)	*Sequence* (σ)	*Value* (ν)
0	1 sec.	1 update	1%
1	10 sec.	15 updates	5%
2	40 sec.	100 updates	30%
3	2 min.	750 updates	60%
4	5 min.	1000 updates	90%

Table I
CONSISTENCY VECTOR VALUES FOR EACH CONSISTENCY ZONE

- *Consistency Zone 0*: Regions with the same information (ex: the same *subsubsection*).
- *Consistency Zone 1*: Regions with much in common (ex: adjacent *subsubsections*).
- *Consistency Zone 2*: Regions sufficiently related for a relatively frequent update (ex: containing *section*).
- *Consistency Zone 3*: Regions probably unrelated (ex: containing *chapter*).
- *Consistency Zone 4*: Regions certainly unrelated (ex: *preamble*).

B. System Architecture

The system supports concurrent editions in the document regardless of the order in which they happen. This implies the absence of any lost updates due to conflicting updates or any divergence between the local replicas of clients.

The distributed system that supports the cooperation group is composed by an aggregate of client nodes and a single server node. An interesting aspect about this configuration is that all VFC reasoning is confined to the server. This has the great advantage of hiding the VFC behavior from the clients, which highly simplifies the adaptation of applications. Also, this way, we have a node with an updated view of the whole document and consequentially of its structure. This will result in more accurate distance measurements when comparing this solution to one where VFC reasoning is distributed for all clients.

1) Client Nodes: Client nodes are responsible for proactively sending their document updates to the server as soon as they happen. Also, since VFC is a location-aware model, clients also have to tell the server their editing location in the document. This may happen in a explicitly way, through a special-purpose message or implicitly, if the server extracts the edition location from document update messages. This duality gives a flexibility in the client implementation since the adapted application framework is not required to provide position change signals.

An interesting aspect of the VFC adaptation is that, as far as the client nodes know, there is no special technique filtering which and when each client should receive the new updates. Because of this unawareness of the VFC model, client applications suffer very little or no changes to be able to adapt to the VFC mode. As seen in Figure 2, if we choose not to use explicit location messages, the client architecture remains the same (Figure 2(b)). If on the other hand, we have implicit and explicit location messages we may have to adapt the client (Figure 2(c)). Since we were working with an open source application and we wanted to have a complete VFC enabled scenario, in this work we have both implicit and explicit location messages.

2) Server: Server nodes are responsible for the management of entrance and exit of clients in the cooperation group. Additionally, the server has to send document updates to the clients in the group. In a total consistent system, the server acts more like a propagation gateway. But in a VFC powered system, like this one, the server can also decide to store the updates and delay their propagation.

In this architecture, the VFC reasoning is limited to the Client component. There is a Client component for every member of the cooperation group. Each one is responsible for enforcing VFC limits for the correspondent member. Thus, Client holds all the undelivered updates and controls the user location in the document structure.

3) Data Representation: The VFC model requires a great flexibility in the scheduling of updates. On the other hand, since updates might be reordered, there is an increased risk of finding update conflicts. To deal with these issues, the replicated document is a *Commutative Replicated Data Type* (section II-D). This way, we avoid the use of any complex concurrency control providing at the same time the freedom of edition coveted for this work. In particular, all replicas in the system hold the document in form of a *TreeDoc* [11], [10].

Additionally to the basic definition of the TreeDoc representation, in this work we performed an optimization that results in a compact representation of the TreeDoc which we call a *Partially Expanded TreeDoc*. We also propose an optimization to solve the *Tail problem* which was used in this work only to merge operations.

Partially Expanded TreeDoc: The TreeDoc original definition requires a node in the document tree for every character. Consequentially, as the document size grows, generating the complete tree and sending it to every new client can become tremendous time and space consuming chores. To avoid this problem, we implemented a *partially expanded TreeDoc* representation, which is one where we have non-expanded nodes holding the contents of a certain number of (normal) nodes in a plain string representation. Then, only if required, that node will be expanded. However, even then, the node's children might still be represented efficiently. One other advantage of using this compact representation is that it enables entire operations to be represented as simple and smaller ones.

The Tail Problem: A serious problem that happens after sometime of editing the document without rebalancing the tree (with a *flatten* & *explode* operation), is one that comes from the fact that usually, users producing text, do it by writing the characters one by one. This insertion pattern causes what we called in this work as the *Tail Problem*, which is the creation of a great number of nodes with only the right child, or as we call it in *tail form*. The biggest issue with this tail problem is that the maximum depth of the tree increases much faster than it should which will influence the size of all position IDs (tree paths) in that branch. To solve this problem, the insertion of nodes, when in these conditions, will cause a contraction to a tail-formed node. Then, like in the previous optimization, whenever necessary this tail-formed node will expand in the required position. This solution can generate great savings in used memory which are very useful when propagating the tree to new clients. These savings happen since, instead of having to send all nodes in tail form one by one, we only send one node with all the contents.

C. VFC Enforcement

Server nodes are responsible for enforcing the VFC model. In particular, it is up to the Client component to guarantee that as soon as the VFC limits of some document region are reached, all the undelivered updates in that region are immediately propagated.

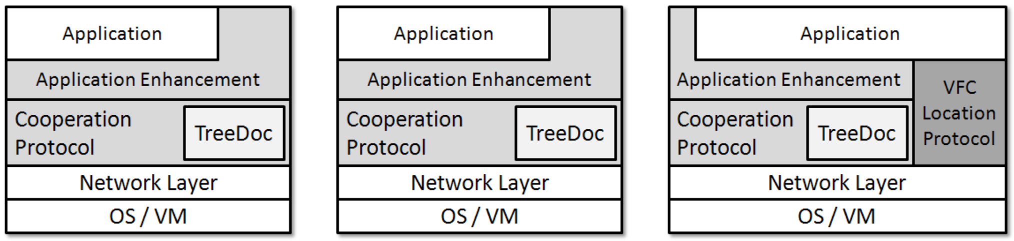
Note that, some events, (e.g. changes in the document structure) can lead to the sudden and unpredictable disrespecting of the VFC divergence limits. However, since there is no way to prevent (or even foresee) these events or their effects, what we intend to guarantee with the designed techniques is that: *the VFC limits are always respected before and after any update*.

The enforcement of VFC limits can be simplified to the handling of these pivotal events:

- Arrival of a New Operation (Update)
- User Location Changed
- Document Structure Changed
- Consistency Zone Timeout

IV. Evaluation

The adaptation of the VFC model was evaluated in a qualitative (section IV-A), a quantitative (section IV-B) and a comparative (section IV-C) perspectives. In the first we argue about the success of the adaptation to the context of the



(a) Total Consistency version (b) VFC version without explict location messages (c) VFC version with explict location messages

Figure 2. Client adaptation Layered Architecture

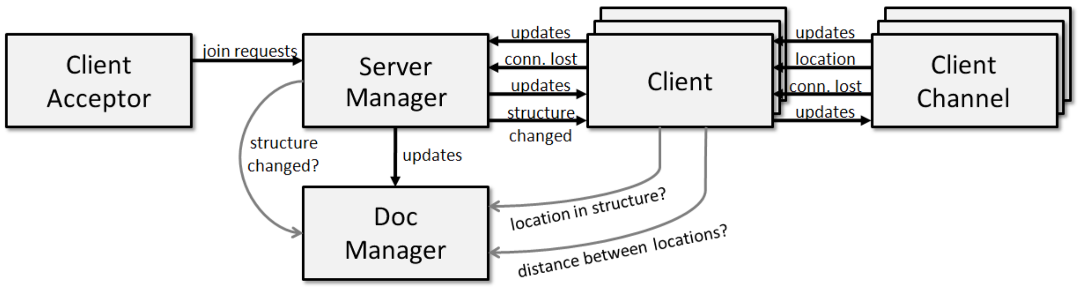


Figure 3. Server node components

document edition. The second consists in a study of the used network resources. The third and last is an argumentative comparison between the use of the VFC model and a Total Consistency (TC) model.

A. *Qualitative Evaluation*

In this work we improved a total-consistent cooperative version of a Latex editor (*Texmaker*) to provide it with the benefits of the VFC model. The main objective of this qualitative study is to assess if the VFC model had a positive impact in the user interaction with the cooperative application.

1) Evaluation of the Application Usability: A document editor using a VFC model has obvious differences when compared to one using a TC model. For instance when a user changes to a previously distant region, he will probably experience a sudden arrival of a series of updates. Also, he is supposed to notice that around his cursor position there is an apparent larger activity rate.

In practice, when using the adapted application, we can experience these exact situations. This gives the user a confidence that there region of edition is as much up-to-date as possible. Also, when testing the application with human users they were able to experience the enforcement of VFC limits and validate their advantages. Finally, there was no noticeable losses to the application performance when compared to the total consistent version. In fact, the types of delays found in the total consistent version were smoothed in the VFC version.

2) Evaluation of the VFC model Adaptation: The adaptation to the VFC model, which was the great objective of this work, was successful. In fact, we were able to provide an enhanced cooperation environment where, using the already described location-aware techniques, there is a selective scheduling of updates according to their probable (or expected) relation (and relevance) with the user point of edition.

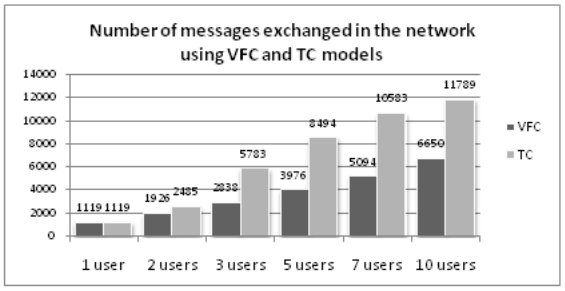
B. *Quantitative Evaluation*

To evaluate the benefits of the VFC model in terms of used resources we conducted a series of tests. With these tests we evaluated the savings in number of exchanged messages, used bandwidth and effectiveness of the merge operations. Also we look at the average time, sequence and value measurements for the sent updates.

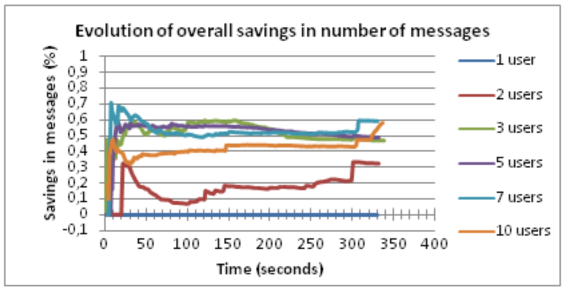
1) Number of Messages: In this work, as far as we can conclude from the experimental results, we were able to have overall reductions in the number of exchanged messages of about 50% when with an already interesting number of users, as can be seen in Figure 4. Of course, when we only have 1 user, we do not have any gains.

Another interesting conclusion was that with these savings, the impacts of increasing the number of users from 3 to 10 users were very little in terms of number of exchanged messages.

In the results we see that from the moment that we have more than two users, that we start having much greater savings in the number of exchanged messages. Also, when



(a) Comparing VFC and TC models



(b) Evolution of overall savings

Figure 4. Number of messages exchanged in the network

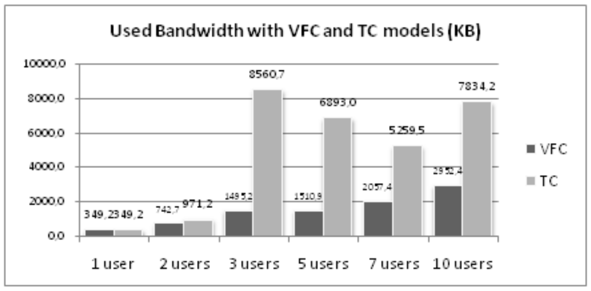
editing a document with only two users, the evolution of the overall savings is less predictable than with another number of users. This can be justified by the great impact that changes in a user position have in all the distances in this type of scenario.

Finally, a comment about the sudden increase in the overall savings in exchanged messages little after the 5 minutes. This increase, coherent with the expected trigger time of the timeout signal for Consistency Zone 4, shows the beneficial effect of the delay and merge of operations.

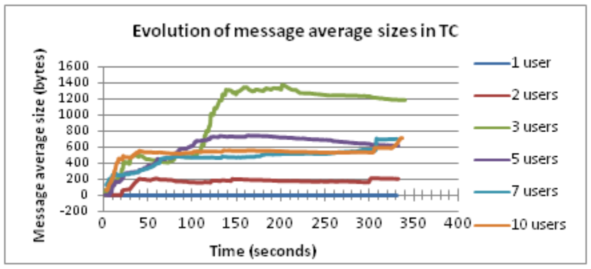
2) Used Bandwidth: The results obtained in these series of tests in terms of used bandwidth are deceiving, and apparently contradicting with the results for the number of exchanged messages. These odd results, seen for example in the test with 3 users 5(a), can be explained by moments in which there was an inflation of the path sizes due to the tail problem (section III-B3) and, consequently, in the size of message (5(b)).

Even though the *tail problem* had so much impact in the results for the Total Consistent model, we can see in Figure 5(a) that, with the VFC model, we were able to damp those effects and control the used bandwidth.

As seen by the savings in the overall used bandwidth, for the tests involving more than two users, we were able to have final savings of around 70%. During the test, since the very first seconds, the overall savings were never below 50%.



(a) Comparing VFC and TC models



(b) Evolution of message average sizes in TC

Figure 5. Bandwidth savings in the network

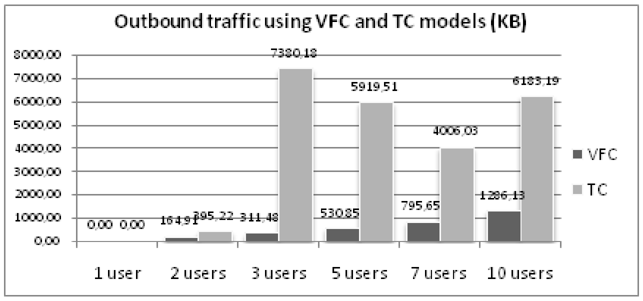


Figure 6. Outbound traffic merge savings

3) Operation Merge Results: The VFC model generates benefits when updates are delayed and merged with each other before being sent. Thus, we now show how efficient were those merge operations, which affected all and only the server outbound traffic.

Looking at the results in Figure 6 we see that the merge operations were able to compress the operations to 90% of the original size, which shows the real potential of the use of the VFC model.

Additionally, these results show the efficiency of the solution used to solve the tail problem and how that solution can be used to generate great savings in the memory required to store a document in form of a TreeDoc.

4) VFC Limits: In table II, we can see the measurements of the average *time* (imposed delay for each operation),

Average VFC measurements					
Zone	**#Regions**	**#Ops**	**Time**	**Sequence**	**Value**
0	398	684	0,34s	1,72 un.	0,48%
1	42	989	1,93s	23,55 un.	7,67%
2	4	369	9,97s	92,25 un.	20,68%
3	3	98	20,14s	32,67 un.	19,43%
4	7	465	259,85s	66,43 un.	26,44%
ALL	454	2605	49,38s	5,74 un.	1,85%

Table II
AVERAGE VFC MEASUREMENTS

sequence (number of operations) and *value* (ratio between the changes and the region size) of all the updates in a session of 5 users during 5 minutes.

The first conclusion is that the updates in the Consistency Zone 0, i.e. around the user location of edition, take an average of 0,34s to be transmitted, which is sufficiently fast no to hinder the usability of the application. Also, about the average values for the time measurements, we can see that they are much below the maximum limits. This means that, even if there is a temporary situation with a peak in the number of incoming updates, it can be rapidly compensated. Another interesting fact is that only the average delay for the Consistency Zone 4 comes close to its maximum limit. This can be explained by the fact that it is not possible to have structural changes causing sudden reach of VFC limits in this zone.

Notice that the number of updated regions in the outermost consistency zones is smaller then predicted. For this reason, the variance of the average value measure is not significant. These results can be easily explained: the bigger the divergence limits, the bigger the time elapsed before the operation is sent. During that time, the user can change his position, which changes the consistency zones and may result in the need to immediately propagate the updates. Even if that does not happen, the longer the update is retained, the greater is the probability of having updates cancelling each other, which will delay even more their sending. Also, their average sequence and value measurements are way below the limits (except in one case) which means that eventually the maximum time limit was reached and the updates were sent. This further demonstrates the savings due to VFC.

C. Comparative Evaluation

In the previous sections we shown the results of several conducted tests. Those tests proved that the VFC model has great benefits when compared with the TC model. The bigger the delay imposed to the updates, the more savings we were able to generate.

In conclusion, unless the the selective delaying of updates will hinder the user experience, which is not an expected scenario, in comparison to the TC model, VFC has a superior management of network resources and, for that reason, will be able to extend the number of users.

V. CONCLUSIONS

In this paper we have presented a *continuous consistency model* which results from the adaptation of the *VFC* model to the scenario of cooperative work, namely of the cooperative edition of documents. The VFC model, besides providing a continuum between strong and weak consistency, is combined with the notion of *locality awareness*, which provides an enhanced cooperation experience where the user is selectively updated in accordance with the contents under edition.

To begin this work, we overviewed the state of the art of two important fields: *consistency maintenance in distributed systems* and *cooperative work tools*.

Then, and most importantly, we defined the architecture for a system powered with the adaptation of the VFC model. Here, we defined the required adaptations that, in our opinion, are required to best adjust the VFC model to the new scenario of cooperative work, specifically document-based cooperative work. In this adaptation, we redefined the concepts of the *user's location* (*pivot*), *distance* between pivots and replicated objects, and *consistency zones*. In this case, we supported the definitions on the notion of *document structure*. Finally, the adaptation also involved porting the update representation from a state-transfer to an *operation-transfer* solution. In fact, to provide the freedom of edition desired in such context, the cooperation was based on a *CRDT*, namely on a *TreeDoc* representation.

The defined architecture was implemented on top of a totally consistent system, also developed in this work. After the implementation, using a number of criteria, we analyzed the success of this work. Other than showing the correctness of the adaptation, the results showed the great potential of the VFC model. In comparison to the use of the TC model, we were able to achieve large reductions in the use of network resources. As such, a VFC powered system provides the enhanced cooperation scenario that we cannot find in the most popular document editors.

In the future, we plan to address the following subjects: the mixture between state-transfer and operation-transfer solutions to achieve more efficient update representations; the fully distribution of the VFC model, where every node has a similar role, and the VFC reasoning is itself carried out in a distributed manner; the introduction of intra-document references in the document structure to shorten the distances between structural distant but semantically related regions; cooperation for entire projects instead of for single files; expand the VFC model to other cooperative environments, namely the cooperative edition of spreadsheets and presentations.

REFERENCES

[1] S. Noël and J.-M. Robert, "Empirical study on collaborative writing: What do co-authors do, use, and like?" *Comput. Supported Coop. Work*, vol. 13, no. 1, pp. 63–89, 2004.

[2] G. Oster, P. Urso, P. Molli, and A. Imine, "Data consistency for p2p collaborative editing," in *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported*

cooperative work. New York, NY, USA: ACM, 2006, pp. 259–268.

[3] S. Weiss, P. Urso, and P. Molli, “Wooki: a p2p wiki-based collaborative writing tool,” in *WISE’07: Proceedings of the 8th international conference on Web information systems engineering*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 503–512.

[4] J. C. Morris, “Distriwiki:: a distributed peer-to-peer wiki network,” in *WikiSym ’07: Proceedings of the 2007 international symposium on Wikis*. New York, NY, USA: ACM, 2007, pp. 69–74.

[5] G. Oster, P. Molli, S. Dumitriu, and R. Mondejar, “Uniwiki: A collaborative p2p system for distributed wiki applications,” in *WETICE ’09: Proceedings of the 2009 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 87–92.

[6] N. Santos, L. Veiga, and P. Ferreira, “Vector-field consistency for ad-hoc gaming,” in *Middleware ’07: Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware*. New York, NY, USA: Springer-Verlag New York, Inc., 2007, pp. 80–100.

[7] Y. Saito and M. Shapiro, “Optimistic replication,” *ACM Comput. Surv.*, vol. 37, no. 1, pp. 42–81, 2005.

[8] H. Yu and A. Vahdat, “Design and evaluation of a continuous consistency model for replicated services,” in *OSDI’00: Proceedings of the 4th conference on Symposium on Operating System Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2000, pp. 21–21.

[9] J. P. Barreto, J. Garcia, L. Veiga, and P. Ferreira, “Data-aware connectivity in mobile replicated systems,” in *MobiDE*, 2009, pp. 9–16.

[10] N. Preguiça, J. M. Marques, M. Shapiro, and M. Letia, “A commutative replicated data type for cooperative editing,” in *ICDCS ’09: Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 395–403.

[11] M. Shapiro and N. Preguia, “Designing a commutative replicated data type,” Computer Science Dept: University of Copenhagen, Tech. Rep., 2007.