Efficient filtering for massively distributed video games

Luciana Arantes, Maria Potop-Butucaru, Pierre Sens, Mathieu Valero

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Efficient filtering for massively distributed video games

Luciana Arantes — Maria Gradinariu Potop-Butucaru — Pierre Sens — Mathieu Valero

N° 7008

Août 2009

ISSN 0249-6399 ISRN INRIA/RR--7008--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
PARIS - ROCQUENCOURT

Efficient filtering for massively distributed video games

Luciana Arantes , Maria Gradinariu Potop-Butucaru , Pierre Sens , Mathieu Valero *

Thème : Réseaux, systèmes et services, calcul distribué
Équipe-Projet Regal

Rapport de recherche n° 7008 — Août 2009 — 19 pages

Abstract: Distributed R-trees overlays recently emerged as an alternative for efficiently implementing DHT-free publish/subscribe communication primitives. Overlays using R-tree index structures offer logarithmic delivery garanties, guarantee zero false negatives and considerably reduce the number of false positives. In this paper we extend the distributed R-trees (DR-Trees) in order to meet two key requirements in massively distributed video game applications: load balancing and low latency. Our optimizations target both the structural organisation of the DR-Trees and the publication policies. The contribution of the current work steams in an extensive evaluation of the novel structure along four parameters: latency, load, scalability and the rate of false positives. Interestingly, the novel structure performs better than the traditional distributed R-tree both in terms of load balancing and latency. Additionally, it does not alter the performances related to the scalability and the rate of false positives and negatives a node has to filter.

Key-words: Publish/subscribe, Distributed R-Trees, Performance evaluation, Distributed multiplayer games

* LIP6 - University of Paris 6 - INRIA

Centre de recherche INRIA Paris – Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex
Téléphone : +33 1 39 63 55 11 — Télécopie : +33 1 39 63 53 30

Filtrage d'informations efficace pour les jeux vidéo massivement distribués

Résumé : Les réseaux logiques basés sur des R-trees répartis ont récemment émergé comme une alternative aux DHTs pour implémenter efficacement des primitives de communication de type publication/abonnement. Ces réseaux utilisent les structures d'index des R-Trees pour garantir une délivrance des messages logarithmique, l'absence de faux négatifs tout en réduisant considérablement le nombre de faux positifs. Dans ce rapport nous étendons les R-Trees distribués (DR-Tree) pour les adapter à deux besoins clés des jeux massivement répartis : la répartitions de charge entre les pairs et une faible latence de communication. Nos optimisations modifient à la fois la structure du réseau de pairs et les stratégies de publications. Une de nos principales contributions consiste à évaluer de manière précise la nouvelle structure selon quatre métriques : la latence, la charge, l'extensibilité et le taux de faux positifs. Nous montrons que nos optimisations permettent d'améliorer notablement la répartition de charge et les latences de publication. De plus, elles n'affectent pas les bonnes propriétés des DR-Trees en termes d'extensibilité et de taux de faux positifs et négatifs que chaque pair doit filtrer.

Mots-clés : Publication/abonnement, R-Trees distribués, Evaluation de performance, Jeux multi-joueur répartis

1 Introduction

Publish/Subscribe primitives imposed themselves as novel and efficient communication abstractions with a broad class of applications (e.g. stocks management or communication abstractions for large scale systems). Recently, publish/subscribe primitives found an interesting application in massively distributed video games where the pertinent information has to be efficiently distributed to the interested parties only. In these systems the amount of information a node has to process is critical since nodes have to conserve their computational power and bandwidth in order to fully satisfy the users expectation. Therefore, communication primitives targeted to reduce noisy events (false positives or negatives) are highly requested. Publish/Subscribe implemented on top of distributed R-trees overlays, first introduced in [7], are proven to be efficient communication primitives. They have been designed to offer zero false negatives and reduce the number of false positives. Interestingly, they also offer a logarithmic delivery complexity. These characteristics make them appealing for applications like P2P video games where nodes have to process only pertinent information. However, their main drawback is their unbalanced load. That is, nodes in charge of the top levels of the overlay have to deal with an important load due to the high trafic they have to process (new subscriptions and events are generally filtered using a top-down strategy). Therefore, in P2P video games where the maintenance of the overlay is performed by the players themselves[1], these overlays need to be optimized along two other important criteria: load and latency. The aim of this paper is to improve distributed R-tree in order to offer load balancing and low latency while maintaining their original features related to reduced number of noisy events.

Our contribution. In this paper we optimize the distributed R-Tree overlays in order to meet the two main requirements of massively distributed video games. First, pertinent information has to be quickly distributed to all the interested parties. Second, the residual trafic has to be minimized in order to not penalize users with reduced bandwidth. Our optimizations are twofold. First we target structural optimization duplicating the virtual links between nodes in the distributed R-tree. Then we propose novel strategies for events dissemination that fully exploit the new added links. The real contribution of the paper steams in the extensive evaluation of the performances of our optimized publish/subscribe communication primitive targeting two main criteria: latency and load. Interestingly, the new structure performs better than the traditional distributed R-tree both in terms of load balancing and latency. Additionally, it does not alter the performances related to the structure scalability and the rate of false positives and negatives a node has to filter.

2 Related Work

Publish/subscribe systems have received much attention and have been extensively studied in the last few years [2, 16]. In such systems, consumers specify *subscriptions*, indicating the type of content that they are interested in, using

[1] Note that in these systems players are mainly concerned with their bandwidth and fast reactivity.

some predicate language. For each incoming message (*event*), a *content-based router* matches the message contents against the set of subscriptions to identify and route the message to the set of interested consumers. Therefore, the consumers and the producers are unaware of each other and the destination is computed dynamically based on the message contents and the active set of subscriptions.

Traditional content routing systems are usually based on a fixed infrastructure of reliable *brokers* that filter and route messages on behalf of the producers and the consumers. This routing process is a complex and time-consuming operation, as it often requires the maintenance of large routing tables on each router and the execution of complex filtering algorithms (e.g., [3, 11, 15]) to match each incoming message against every known subscription. The use of summarization techniques (e.g., subscription aggregation [8, 10]) alleviates those issues, but at the cost of significant control message overhead or a loss of routing accuracy.

Another approach to content routing is to design it free of broker infrastructure, and organize publishers and consumers in a peer-to-peer overlay through which messages flow to interested parties. Several designs of DHT-based peer-to-peer publish/subscribe systems were proposed [9, 24, 17, 13, 12, 4, 23]. The main advantage of these approaches is their scalability, although most of them suffer from two problems: the loss of accuracy (apparition of false negatives or false positives) and poor latency in scenarios with high churn. In this paper we are interested in publish/subscribe communication primitives that meet the massively distributed games requirements: reduce number of noisy events, load balancing and low latency. Hence, for such approaches to be efficient, the overlay on top of which the primitive is implemented must: avoid *false negatives* (a registered consumer failing to receive a message it is interested in); minimize the occurrence of *false positives* (a consumer receiving a message that it is not interested in); *self-adapt* to the dynamic nature of the systems, with peers joining, leaving, and failing; *balance the load* of the subscribers in charge of the overlay maintenance and efficiently distribute events to the interested parties (provide a low latency). None of the previously mentioned systems meet all these criteria.

In the massively distributed video games the most popular publish/subscribe system is Mercury having a similar design with [23]. Mercury [6] is a peer-to-peer DHT supporting multi-attribute range-queries and explicit load balancing on top of which a First Person Shooter (FPS) dedicated publish/subscribe has been built and used in Caduceus [6] and Colyseus [5]. Subscriptions are mapped on range queries, publications on classic DHT *put*() operation and each attribute to a dimension. Mercury creates one ring per dimension; each peer belongs to several rings. It doesn't scale with dimension number however it performs well in systems with moderated number of dimensions. Each peer knows for each ring its predecessor, its successor and has k long links obtained by lazy random-walk. k may vary from one peer to another, from one node to another and from one ring to another. On publication, an event is inserted in each ring where it is routed according to the corresponding attribute (resp. dimension). Under the assumption of uniform node's ranges on each ring, Mercury route any event in $O(\frac{1}{k}log^2 n)$. Due to the ring-overlay design nodes in Mercury have to process both false positives and negatives.

Kademlia [19] is a DHT that has been used as an underlayer for a *Second Life* peer-to-peer client [22]. Peers are connected as a BST according their

160-bit IDs. For each bit of its ID, a peer keeps a list of k peers having a different value for that bit. k is parameterisable, it may differ from one peer to another and from one bit to another. Those links allows $O(log\ n)$ publication. Moreover, peers heavily cache information about other peers during routing process. As routing is done according XOR metric, it takes high benefits of caching. But DHT API is inadequate for localisation system that express range queries or zone of interests [21]. Walkad [21] is an extension of Kademlia that tends to satisfy those requirements by organizing the Kademlia keyspace in a reverse binary trie using the Gray Code. It divides the virtual world into atomic disjointed areas (cells) and associates a key to each of them. It makes adjacent cells having adjacent keys, emulating spatial locality and region containment.

3 Publish/Subscribe Model

We consider a distributed dynamic system where publishers and subscribers are organized in a broker-free overlay, i.e., a peer-to-peer structure. Hence, every peer in the overlay may have three roles: publisher/subscriber and router. Each peer typically registers one subscription and may or not publish events. Also, the peers may participate in the event dissemination, i.e., the event matching and forwarding process is completely distributed among the peers in the system.

As most other publish/subscribe systems, we assume that an event contains a set of *attributes* with associated values. In this work we consider complex filters expressed as the conjunction of multiple range predicates. Geometrically, these complex filters define poly-space rectangles in an Euclidean space. This representation captures well the range filters expressed in most popular publish/subscribe systems (e.g., [2, 20, 8, 14]).

An event specifies a value for each attribute and corresponds geometrically to a point. Without restraining the generality, we illustrate our algorithms on two-dimensional filters corresponding to rectangles in a two-dimensional space. If one attribute is undefined, then the corresponding rectangle is unbounded in the associated dimension. If an attribute is composed of disjoint ranges, the subscription will be represented as multiple rectangles. In that case, we can split the original subscription into several new subscriptions, one per rectangle, or merge the multiple ranges of every attribute to produce a single subscription, at the price of degraded accuracy.

In order to improve event dissemination, publish/subscribe systems can take advantage of the property of *subscription containment*,[2] which is defined as follows: subscription S_i contains another subscription S_j (written $S_i \sqsupseteq S_j$) iff any event m that matches S_j also matches S_i. Conversely, we say that S_j is contained by S_i and we write $S_j \sqsubseteq S_i$. Note that the containment relationship is transitive and defines a partial order. Geometrically, subscription containment corresponds to the enclosure relationships between the poly-space rectangles. When organizing the peers based on the containment relationship of their subscriptions, only the peers that are interested in an event will participate in the matching and forwarding procedure. In this way, events can be quickly disseminated without incurring significant filtering cost.

[2] The term *covering* is also commonly used in the literature.

4 R-Trees Overlays

In this section we recall the main caracteristics of the R-Tree index structure and its distributed version.

4.1 R-Trees index structures

R-trees were first introduced in [18]. An R-tree is a height-balanced tree handling objects whose representation can be circumscribed in a poly-space rectangle. Each leaf-node in the tree is an array of pointers to spatial objects. An R-tree is characterized by the following properties:

- Every non-leaf node has a maximum of M and at least m entries where $m \leq M/2$, except for the root.
- The minimum number of entries in the root node is two, unless it is a leaf node. In this case, it may contain zero or one entry.
- Each entry in a non-leaf node is represented by (mbr,p), where the mbr is the *minimum bounding rectangle* (MBR) that encloses the MBRs of its child node and p is the pointer to the child node. Each entry in a leaf node is represented by (mbr,oid), where the mbr is the MBR that spatially encloses the object and oid is the pointer to the object.
- All the leaf nodes are at the same level.
- The height of an R-tree containing N objects is $\lceil log_m(N) \rceil - 1$.
- The worst space utilization for each node except the root is m/M.

In a classical R-tree structure, the actual objects are only stored in the leaves of the tree and the internal nodes only maintain MBRs.

4.2 Distributed R-tree Overlay

In this section we recall the characteristics of the DR-trees index structure introduced first in [7]. Subscribers self-organize in a balanced virtual tree overlay based on the semantic relations between their subscriptions. Each filter is a rectangle and can be represented using coordinates in a two dimensional space. The overlay preserves the R-trees index structure features: bounded degree per node and search time logarithmic in the size of the network. Moreover, the proposed overlay copes with the dynamism of the system.

Unlike the traditional R-trees, each node in the structure is under the responsibility of a peer. The DR-tree structure is defined by the logical links between subscribers or peers depending on the relation between their filters. Every peer in the overlay registers for at least one subscription that is stored at the leaves of the tree. Depending on the nature of a peer's subscription, it may be responsible also for internal nodes of the tree. The subscriber responsible for an internal node of the tree filters events for all subscribers responsible for the nodes in its subtree. In order to maintain the balanced nature of the tree, a subscriber responsible for some node in the overlay is also responsible for one internal node at each level of its subtree. More precisely, an internal node p is

recursively its own child in the subtree rooted by p. Therefore, a peer may have to maintain more than one parent link and children set.

The organization of the subscribers has a strong influence on the routing accuracy and the number of false positives in the system. The following property is preserved:

Property 1 (Weak Containment Awareness)**.** *Given two filters S_1 and S_2 with $S_1 \sqsubseteq S_2$, then the topmost instance of S_1 is not an ancestor of the topmost instance of S_2 in the DR-tree.*

This property guarantees that a containee filter will not be a parent of a container filter, as it would degrade routing accuracy.

In addition, it is desirable to implement a stronger variant of the containment awareness property:

Definition 1. *Let filter S_1 be called an* accessor *of filter S_2 if the topmost instance of S_1 is an ancestor or sibling of the topmost instance of S_2 in the tree.*

Property 2 (Strong Containment Awareness)**.** *Given two filters S_1 and S_2 with $S_1 \sqsubset S_2$, then either S_2 is an accessor of S_1 in the DR-tree, or there exists S_3 such that $S_1 \sqsubset S_3$, $S_2 \not\sqsubseteq S_3$, $S_3 \not\sqsubseteq S_2$, and S_3 is an accessor of S_1 in the DR-tree.*

This property would ensure that a containee filter is a descendant of its containers. Because of the height-balancing mechanism, it might not be possible to register a containee deep enough in the tree as child of one of its container; in that case, it can be inserted as a sibling of the container. The second clause of the property deals with the case of a filter having two container filters that do not cover each other (remember that the containment relationship is a partial order). Therefore, the containee may become a descendant of either of its container.

In order to preserve the containment awareness properties and minimize the likeliness for false positives, the root of a subtree is the node whose current MBR is largest, i.e., which provides most coverage over the MBR of the new root.

5 Optimized Distributed R-Tree

In this section we detail the optimizations we propose for the classical distributed R-trees described in Section 4. We address both the topological extensions and propose novel publication strategies. In the following node refers a peer in the DR-tree overlay.

5.1 Topological extensions

In order to improve communications between peers (according different criteria such as availability, latency and load balancing) we improve the connectivity of a DR-Tree by adding some links to the communication graph[3]: connections between brother nodes and ancestors.

[3]The graph defined by the virtual connections between the peers in the DR-Tree overlay

Brothers Connections. In order to reduce the delivery time of events and hence the latency of events distribution we add links between brother nodes. Logically, two peers are brothers if and only if they share the same father. This relation is symmetric, transitive and non-reflexive. Note that other overlays such as [23] use the brother relation in order to connect similar nodes. In most cases brother nodes form a ring or a multi-ring. In our case, for the sake of efficiency, the brother relation is a crossbar. This structure offers the maximal performance in terms of latency since messages within the brother set are routed in one hope. The maintenance costs are low since the number of brothers per node does not exceed M nodes[4].

Ancestors Connections. A node is the ancestor of another node if and only if the former is the father of the latter or the father of an ancestor of the latter. In the classical Distributed R-trees, subscribers learn their ancestors during their connection or the routing process. Interestingly, this information has not been exploited so far. In the following, we consider each node has a specific field where it stores its ancestors.

5.2 Publishing strategies

We denote that a local event is an event that has been published by the node itself, that an upgoing event is an event that a node has received from one of its children, and that a downgoing event is an event that a node has received from its father. A publishing strategiy thus defines the traffic rules, i.e., the routes that local, upgoing and downgoing events should take.

In the classical DR-trees the publication always starts from the root of the tree. However, the containment relation between MBRs entails the filtering in both directions. Roughly, a peer receiving an event has to forward it to its interested children and eventually to its father. In the following we enrich the publication policy with four novel strategies.

5.2.1 Double wave strategy

Intuitively, in this strategy, an event is send up until it reaches the root, then it is sent down towards interested peers (Figure 1). Local events are always forwarded to the publisher's father. Upgoing events received by a non root peer are forwarded to it's father. The root sends these events to its interested children. Downgoing events are always forwarded to the interested child.

5.2.2 Enhanced double wave strategy

The enhancement consists in an earlier start of downward propagation (Figure 2). When an internal peer receives an upgoing event from one of its children, it forwards the event both to its other interested children and to its father. In this sense, an internal peer behaves like a "local root", initiating the second wave in its subtree.

[4] Note that in [7] M equals 20 has been proven to be a good compromise between the cost of maintenance and the number of false positives a node receives

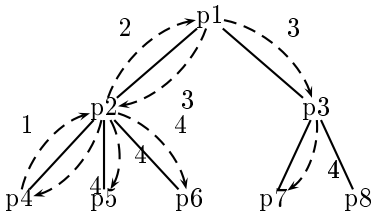


Figure 1: Double wave strategy: $p4$ publishes an event that is of interest of every peer

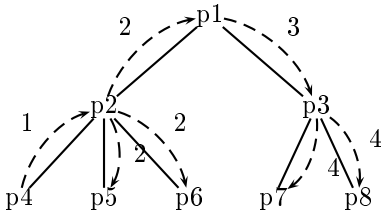


Figure 2: Enhanced double wave strategy: $p4$ publishes an event that interests every peer

5.2.3 Brothers wave strategy

This strategy uses brother links to exploit "tree-locality" of a publication (Figure 3). The idea is that events that interest a node might also interest its brothers as well. Therefore, the publication strategy is the following: local events are forwarded to publisher's interested brothers, children, and father; upgoing events are also forwarded to the receiver's interested brothers and fathers (if the receiver is not the root); downgoing events are forwarded to the interested children.

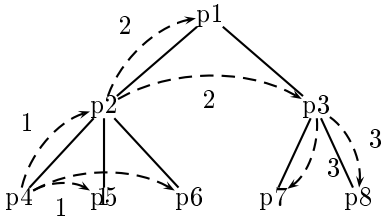


Figure 3: Brothers wave strategy: $p4$ publishes an event that interests every peer

5.2.4 Ancestors wave strategy

This strategy uses ancestors link to maximize messages diffusion parallelization. Local events are published to interested children and every ancestors of the publisher (Figure 4); upgoing and downgoing events are forwarded to interested children.

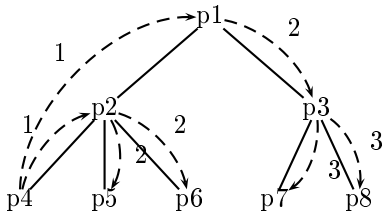


Figure 4: Ancestors wave strategy: $p4$ publishes an event that interests every peer

6 Performances

This section presents a set of results aimed at evaluating the performance of the four published strategies described in the previous section.

6.1 Simulation environnement and parameters

Experiments were conducted on top of PeerSim[1], a Java-based discrete event simulator. They last 600 cycles where a cycle is a discrete unit of time. Publication frequency was 0.5 event per cycle for each peer. Network latency between two peers was 1 cycle with a jitter of $\pm$ 0.1 cycle.

We have considered a 2D virtual area of $[\![0, 1024]\!] \times [\![0, 1024]\!]$ and a network with 1024 peers with one subscriber per peer. Each peer (subscriber) has just one zone of interest, whose height and width are uniformly randomly distributed between $[\![5, 50]\!]$, and one zone of publication. We denote the *covering zone* of a peer the MBR of the uppermost level that it holds.

Every non-leaf node of the DR-Tree has a maximum of M=8 and a minimum of m=4 entries, except the root which has 2 entries. For the sake of evaluation, nodes can be grouped by level: 0 is the root level, 1 is root's children level, and so on. The level of the leaves is equal to the RTree height which is equal to 4 in our experiments.

Subscription distribution: Most of the massively distributed video games present hotspot zones, i.e. "popular" regions in which a group of peers have similar interests. Thus, based on population distributed of existing games, we have considered in our experiments four hotspot distribution configurations for the 1024 peer subscriptions of the system:

- **Cold** (no hotspot): subscriptions are uniformly randomly distributed.
- **Warm** (not very "popular" hotspots): the number of hotspots is $1024/8 = 128$.
- **Hot** ("popular" hotspots): the number of hotspots is $\sqrt{1024} = 32$.
- **Burning** (very "popular" hotspots): the number of hotspots is equal to $log(1024) = 10$ hotspots.

The *Cold* and *Warm* hotspot distributions respectively model the population distribution of deserted zones of DVE and interested zones of FPS games. The *Hot* distribution represents the population distribution of dense zones of

DVE like towns in MMO-RPG (*World Of Warcraft*, *Dofus* or popular islands of *Second Life*) while the *Burning* one maps the population distribution of massive battlefields in MMO-RPG or wide events (concert, meeting).

Publication pattern: Peers (players) subscribe to the geographic area where they are and publish events related to their positions/movements/actions. However, in video games, players are usually interested in a small part of the game map (zone) and they only interact with entities that are in that zone. Such a behavior thus implies that the publication zone of a peer corresponds to its zone of interest, i.e., a peer publishes just in its own zone of interest. To our experiments, we have then considered that publications are uniformly randomly distributed in publishers' subscription zones.

Metrics: As previously explained, our goal in proposing new publishing strategies is to provide both low publication latency and good load balancing without increasing noisy events such as false positives (DR-Tree does not present false negatives) or limiting scalability of the system. Hence, the metrics we have used to evaluate the four strategies are:

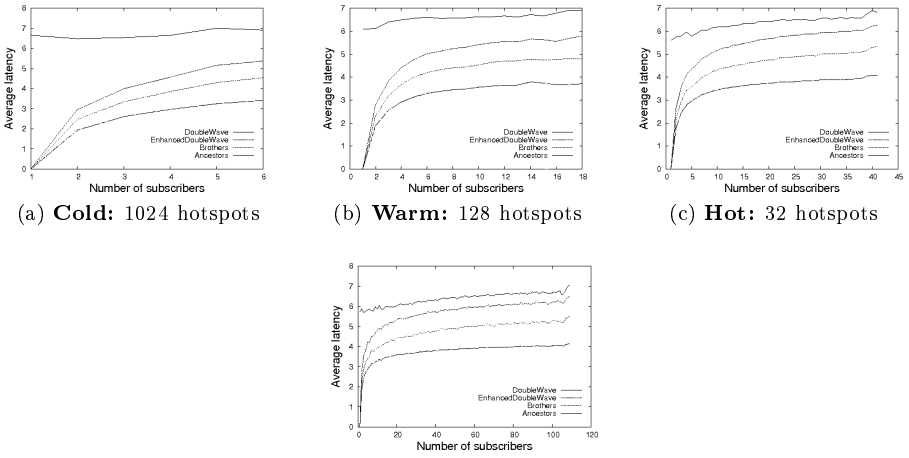
- **Latency**: the average time (in cycles) elapsed between the moment an event is published and its delivery to all subscribers which are interested in it;
- **Message load**: this metric concerns both the *fan in*, the average number of received messages per peer and *fan out*, the average number of sent messages per peer;
- **False positive**: the average number of false positives per level of the DR-Tree.
- **Scalability**: this metric concerns the latency when the number of peers increases.

6.2 Latency

In video games, latency is closely related to interactivity, gameplay smoothness and game experience quality. It measures the elapsed time between the moment an event takes place and the moment all interested players are aware of it (e.g. the time elapsed between a bomb's explosion and the moment every near player is warned of it; the elapsed time between a player kills another one and the moment every witness "sees" this action, etc...).

Figure 5 shows the latency evaluation results for the four publication strategies defined in Section 5. X-axis corresponds to the number of subscribers concerned by a publication. Notice that the colder hotspots are, the lower the number of interested subscribers is. Y-axis corresponds to the average total publication time.

Since a peer publishes in its respective zone of interest, a publication is delivered at least to it. Thus, except for the *Double Wave* strategy where every publish event must be firstly routed to the root, whenever an event is delivered to exactly one peer (the event publisher), the average global publication time is equal to zero independently of the hotspots distribution. On the other hand,



(a) **Cold:** 1024 hotspots (b) **Warm:** 128 hotspots (c) **Hot:** 32 hotspots

(d) **Burning:** 10 hotspots

Figure 5: Average total propagation time versus event popularity

the higher the number of peers concerned by an event, the higher the chances that the event will have to be forwarded far from the the publisher and hence routed through more peers. Therefore, such an event will take more time to be delivered to all interested subscribers.

Compared to the *Double Wave*, *Ancestors*, *Brothers* and *Enhanced double wave* latency gains are quite significant for all hotspot distributions. In particular, *Ancestors* strategy is around 35% better than *Double wave* strategy for the *Burning* distribution, and around 45% better for the *Cold* distribution.

It is worth pointing out that the curves of Figure 5 could be roughly interpreted as "the number of hops versus the number of reachable nodes in the communication graph". Thus, since the communication graph is a tree, whose height is majored by the height of the DR-tree, the curves have a logarithmic behavior. The inflexion point of curves corresponds to the tree's height. Figure 5 shows also that hotspots distribution has small impact on overall latency; in any case, each strategy's curve stabilizes around the same value, which is an interesting result for video games since latency is always a matter of concern for them. Furthermore, the zone of interest of a player is very likely to change during the game but, due to the mentioned stabilization, such a change will probably not affect the game's reactivity.

6.3 Message load

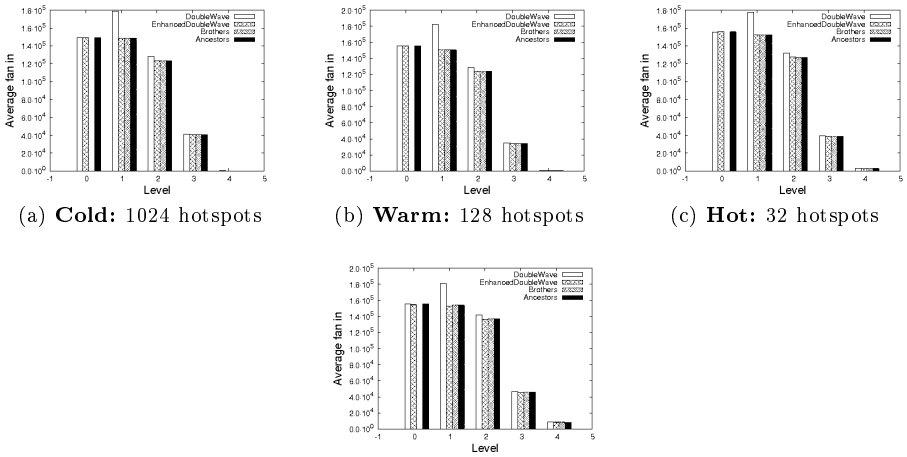
In the previous section we have shown that *Ancestors*, *Brothers* and *Enhanced double wave* provide significant latency gains when compared to *Double wave*. In the following we investigate two metrics *fan in* and *fan out* which are related to the node load. For a given peer, both the *fan in* and the *fan out* are dependent of the following three factors:

- peer's zone of interest
- peer's routing upward activity

- peer's routing downward activity

Note that these factors may have very different order of magnitude according to the publication strategy and the level of the peer in the overlay.

Fan in evaluation: Figure 6 shows some results related to the *fan in* metric. The X-axis corresponds to the R-Tree levels. The leftmost level is the root, the rightmost level corresponds to the leaves, and the in-between levels correspond to internal nodes. For the Y-axis, each bar represents the average *fan in* of peers at a given level. It is worth remarking that standard variation of *fan in* for peers at each level is very low.



(a) **Cold:** 1024 hotspots (b) **Warm:** 128 hotspots (c) **Hot:** 32 hotspots

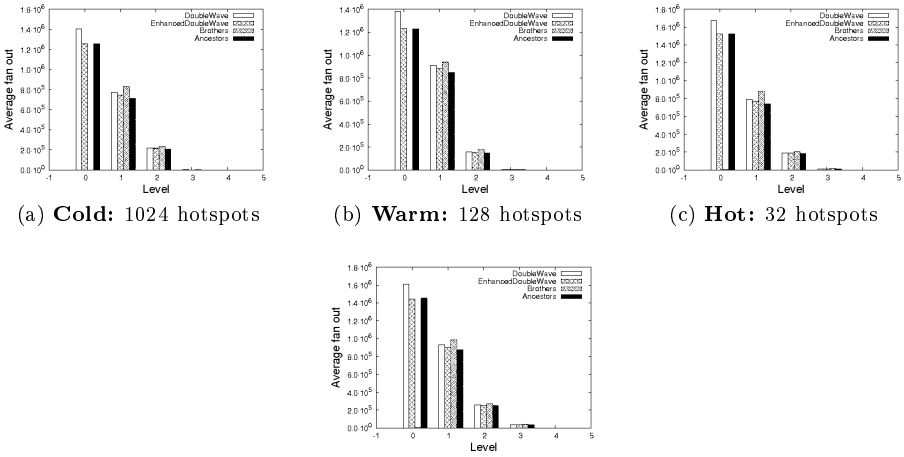
(d) **Burning:** 10 hotspots

Figure 6: Fan in versus level

We can observe in Figure 6 that all strategies (except *Double Wave*) are roughly equivalent in terms of *fan in*, regardless of the hotspot distribution. Since DR-tree routing avoids false negatives, a peer receives an event either if it is interested in the event or if some of its children is. In other words, a peer receives an event only if the latter is in its zone of interest or in its covering zone. The closer to the root a peer is, the large its covering zone is and thus the higher the number of upgoing events it receives which explains why the bars of Figure 6 decreases when the level increases, independently of the hotspots distribution.

An interesting remark is that in the case of the *Brothers* strategy, the root peer *fan in* is equal to 0 and strictly equivalent to a leaf peer for all hotspot distributions. As its children know each other, the root peer is not involved in routing events and thus it receives only those events in which it is interested. On the other hand, in the *Double wave* strategy, internal nodes are more loaded than with other strategies. This happens because these nodes can receive the same event twice: during the first wave when events are only forwarded towards the root peer and then during the second wave when events are filtered towards leaf peers. Such a behavior also explains why the root peer is not the most loaded one as it never receives the same event twice.

Fan out evaluation: Figure 7 presents some evaluation results of the *fan out* metric. Like to the *fan in* figure, the X-axis represents the R-Tree levels. In the Y-axis, each bar corresponds to the average *fan out* of peers at each level. The standard variation of *fan out* for peers of a given level is very low.



(a) **Cold:** 1024 hotspots (b) **Warm:** 128 hotspots (c) **Hot:** 32 hotspots

(d) **Burning:** 10 hotspots

Figure 7: Fan out versus level

Figure 7 confirms that all strategies, except *Double Wave*, are roughly equivalent in terms of *fan out* for all hotspot distributions. Similarly to the *fan in*, the closer to the root a peer is, the higher the number of upgoing events it has to forward to both its father and its children which have interest in them.

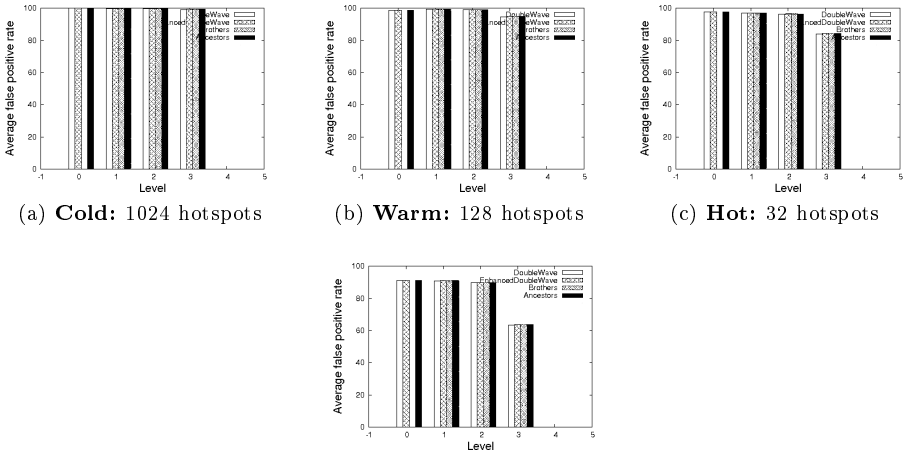
Two points are worth remarking with regard to the *Brothers* strategy. Firstly, the internal nodes are slightly more loaded than with other strategies. The explanation for it is the "horizontal routing" of such an strategy which mostly involves leaves and internal peers in order to reduce the cost of event's upward propagation. Secondly, as already mentioned in the *fan in* evaluation, the root peer is not engaged in the routing of events. Hence, its *fan out* is equivalent to a leaf peer for any hotspot distribution.

6.4 False positive

An event is considered as a false positive by a peer if the latter is not interested in it, i.e., if the event is in the peer's covering zone but not in the peer's zone of interest.

Figure 8 presents our evaluation results related to false positives. X-axis is the levels of the DR-Tree similarly to the *fan in* and *fan out* figures. In the Y-axis, each bar corresponds to the average percentage of false positives for peers of each level. As this metric is highly related to the *fan in*, the standard variation is also very low.

All strategies are equivalent in terms of false positive rate independently of the hotspots distribution. For a leaf peer, the zone of interest and covering zone are equals. Hence, it receives only events in which it is interested, i.e., no false positive occurs. However, the closer to the root a peer is, the wider its covering



(a) **Cold:** 1024 hotspots (b) **Warm:** 128 hotspots (c) **Hot:** 32 hotspots

(d) **Burning:** 10 hotspots

Figure 8: False positive rate versus level

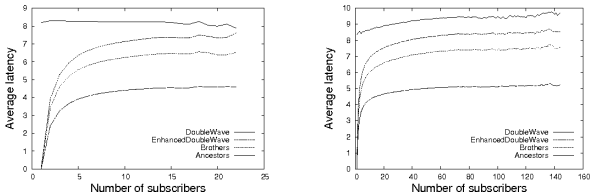
zone is and thus the higher the chances that it receives events that are in its covering zone but not in its zone of interest which leads to higher false positive rates.

We can also observe in the same figure that the overall false positive rate decreases with the popularity of the hotspots since the number of zones of interest that overlap increases as well. The more they overlap, the higher the chances for a peer to receive events that are in its zone of interest which thus leads to slightly lower false positive rate.

A third remark is that in the case of the *Brothers* strategy, root peer behaves like a leaf peer. As explained in the description of this strategy, the root peer only receives events in which it is interested in. Therefore, no false positive occurs.

6.5 Scalability

We have conducted the same set of experiments as the ones shown in Figure 5, but with 10,000 peers instead of 1024. Our results are shown in Figure 9.



(a) **Cold:** 1024 hotspots (b) **Hot:** 128 hotspots

Figure 9: Total propagation time for 10,000 peers

The shape of the curves is quite similar to those of Figure 5 (i.e., similar inflexion points and asymptotic behavior). The 10-times multiplication of peers number results in an increase of the average latency by 25% for all strategies. Such an overhead can be explained since latency is closely related to the communication graph's height which is majored by R-Tree's one which grows logarithmically with peers number. The DR-Tree we have considered in our experiments has a degree of (m=4;M=8) which implies that the height of the tree (and therefore the height of the communication graph) increases when the number of peers grows from 1024 to 10,000. However, an important point to emphasize is that latency gains of *Ancestors*, *Brothers* and *Enhanced double wave* strategies in relation to *Double wave* strategy do not change when the number of peers of the network increases: the former's latencies are around 40% lower than the latter's latency.

7 Conclusion

In this paper, we have proposed some extensions to distributed R-Trees which meet the requirements of distributed video games. In multiplayer games, participants share a single instance of the game but each participating node only needs information relevant to his/her associated player. Publish/subscribe is thus an interesting approach for multiplayer games for filtering information but also for overcoming the problem of scalability caused by centralized client-server architectures or broadcast communication. Despite scalability, publication latency, reduction of noisy events, and load balancing are also essential concerns in distributed games in order to maintain fairness between players and conserve computational power and bandwidth of peers. However, traditional publish/subscribe protocols do not meet these key requirements. To this end, we have proposed in this article some structural modifications of DR-Trees by adding shortcut links in the overlay. Based on the results of extensive evaluation experiments, our paper shows that our novel link structures outperform the traditional DR-Tree both in latency and load balancing of peers. Furthermore, they do not entail more false positives and the system scales well.

References

[1] Peersim: A peer-to-peer simulator.

[2] M.K. Aguilera, R.E. Strom, D.C. Sturman, M. Astley, and T.D. Chandra. Matching events in a content-based subscription system. In *PODC*, pages 53–61, 1999.

[3] M. Altinel and M.J. Franklin. Efficient filtering of XML documents for selective dissemination of information. In *VLDB*, pages 53–64, 2000.

[4] E. Anceaume, A.K. Datta, M. Gradinariu, G. Simon, and A. Virgillito. A semantic overlay for self-* peer-to-peer publish subscribe. In *ICDCS*, 2006.

[5] Ashwin Bharambe, Jeffrey Pang, and Srinivasan Seshan. Colyseus: a distributed architecture for online multiplayer games. In *NSDI*, pages 12–12, Berkeley, CA, USA, 2006. USENIX Association.

[6] Ashwin R. Bharambe, Mukesh Agrawal, and Srinivasan Seshan. Mercury: supporting scalable multi-attribute range queries. *SIGCOMM Comput. Commun. Rev.*, 34(4):353–366, 2004.

[7] S. Bianchi, A.K. Datta, P. Felber, and M. Gradinariu. Self-stabilizing peer-to-peer spatial filters. Technical report, LIP6, Universite Paris 6, 2007.

[8] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. Design and evaluation of a wide-area event notification service. *ACM TOCS*, 19(3):332–383, 2001.

[9] M. Castro, P. Druschel, A.M. Kermarrec, and A. Rowston. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE JSAC*, 20(8):1489–1499, 2002.

[10] C.-Y. Chan, W. Fan, P. Felber, M. Garofalakis, and R. Rastogi. Tree pattern aggregation for scalable XML data dissemination. In *VLDB*, 2002.

[11] C.-Y. Chan, P. Felber, M. Garofalakis, and R. Rastogi. Efficient filtering of XML documents with XPath expressions. *VLDB Journal, Special Issue on XML*, 1(4):354–379, 2002.

[12] R. Chand and P. Felber. Semantic peer-to-peer overlays for publish/subscribe networks. In *Euro-Par*, 2005.

[13] P. Costa, M. Migliavacca, G. P. Picco, and G. Cugola. Epidemic algorithms for reliable content-based publish/subscribe: An evaluation. In *ICDCS*, 2004.

[14] G. Cugola, E. Di Nitto, and A. Fugetta. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. *IEEE Transactions on Software Engineering*, 27(9):827–850, 2001.

[15] Y. Diao, P. Fischer, M. Franklin, and R. To. YFilter: Efficient and scalable filtering of XML documents. In *ICDE*, 2002.

[16] P.Th. Eugster, P. Felber, R. Guerraoui, and A.M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[17] A. Gupta, O.D. Sahin, D. Agrawal, and A. El Abbadi. Meghdoot: Content-based publish:subscribe over P2P networks. In *Middleware*, 2004.

[18] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1984.

[19] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric, 2002.

[20] B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps. Content based routing with Elvin4. In *AUUG2K*, 2000.

[21] Matteo Varvello, Ernst W Biersack, and Christophe Diot. A walkable kademlia network for virtual worlds. In *IPTPS*, Apr 2009.

[22] Matteo Varvello, Christophe Diot, and Ernst W Biersack. P2p second life : experimental validation using kad. In *Infocom*, Apr 2009.

[23] S. Voulgaris, E. Rivire, A. Kermarrec, and M. van Steen. Sub-2-Sub: Self-organizing content-based publish subscribe for dynamic large scale collaborative networks. In *IPTPS*, 2006.

[24] S.Q. Zhuang, B.Y. Zhao, A.D. Joseph, R. Katz, and J. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of the International Workshop on Network and OS Support for Digital Audio and Video*, 2001.

Contents

Centre de recherche INRIA Paris – Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399