مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (39) العدد (3) 2017
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (39) No. (3) 2017

تحسين خوارزمية عضوية المجموعة في منصّة عمل مجموعة الغرض الموزّع 'Jgroup'

د. رضوان دنده*
د. قاسم قبلان**
علي اسماعيل***

(تاريخ الإيداع 12 / 3 / 2017. قُبِل للنشر في 15/ 6 / 2017)

ملخّص

تدمج Jgroup نموذج مجموعة الغرض (Object Group) مع نموذج الغرض الموزع من Java RMI، مزوّدة منصّة عمل (platform) ملائمة لتطوير تطبيقات موزعة موثوقة قابلة للتجزئة، فهي تعتمد تقنية واحدة (RMI) في جميع تفاعلاتها؛ سواء الداخلية لتحقيق التنسيق بين أغراض مجموعة المخدم أو الخارجية اللازمة لاتصال الزبون مع مجموعة الغرض. نظراً لديناميكة الشبكة؛ الناتجة عن *انضمام* مخدمات جديدة إلى مجموعة الغرض و *مغادرة* مخدمات أخرى أو الناتجة عن حدوث حالات *تجزئة* بسبب انقطاع في شبكة الاتصال بين المخدمات، فإن خدمة عضوية المجموعة القابلة للتجزئة في Jgroup تتبّع مسار هذه التغيّرات لتزوّد كل مخدّم بتقرير يسمى منظاراً (view) يحوي قائمة بالأعضاء الحاليين القابلة للاتصال والتنسيق فيما بينها. تتميز هذه الخدمة في Jgroup بأنها تحافظ على استمرارية توفّر الخدمة الموزّعة في جميع أجزاء الشبكة؛ بدلاً من محدوديتها في جزء واحد فقط. عندما يتم دمج الأجزاء بعد غياب التجزئة في شبكة الاتصال، تبني خدمة دمج الحالة من Jgroup حالة عامة متناسقة لتصلح أي انحراف ناتج عن تحديثات متناقضة في الأجزاء المختلفة.

يجب على خدمة العضوية أن تضمن تحميل منظار فقط بعد التوصّل إلى توافق على تركيبه بين جميع المخدمات الموجودة ضمن المنظار(خاصية *التوافق على المنظار*). لهذه الغاية؛ يتم تبادل رسائل تخمين عن المنظار المتوقع بين جميع المخدمات؛ مما يسبّب حمولة زائدة (overhead) عبر الشبكة.

تحسّن هذه المقالة أداء خوارزمية العضوية المسؤولة عن تحقيق خاصية التوافق على المنظار، من خلال السماح لأول مخدم فقط يكتشف حالة التغيّر في العضوية بإرسال تخمينه، بدلاً من قيام جميع المخدمات بذلك.

تبيّن *نتائج* تقييم الأداء أن الخوارزمية المحسنة *تخفّض* عدد التخمينات المرسلة، وتزداد نسبة التخفيض مع تزايد عدد المخدمات المتواجدة ضمن المنظار، وتستغرق الخوارزمية المحسنة بشكل تقريبي الفترة الزمنية نفسها التي تتطلّبها الخوارزمية السابقة للوصول إلى التوافق.

الكلمات المفتاحية: نموذج مجموعة الغرض؛ أنظمة اتصالات المجموعة الموجهة بالمنظار؛ Jgroup؛ خوارزمية عضوية المجموعة.

* أستاذ– قسم النظم والشبكات الحاسوبية– كلية الهندسة المعلوماتية – جامعة تشرين– اللاذقية – سورية.
** مدرّس– قسم النظم والشبكات الحاسوبية– كلية الهندسة المعلوماتية – جامعة تشرين– اللاذقية – سورية.
*** طالب دراسات عليا (دكتوراه) – قسم النظم والشبكات الحاسوبية– كلية الهندسة المعلوماتية– جامعة تشرين– اللاذقية– سورية.

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (39) العدد (3) 2017
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (39) No. (3) 2017

Improvement of Group Membership Algorithm in distributed object group platform 'Jgroup'

Dr. Radwan Dandah[*]
Dr. Kasem Kabalan[**]
Ali Esmaeel[***]

(Received 21 / 6 / 2016. Accepted 21 / 1 / 2017)

□ ABSTRACT □

Jgroup integrates the object group paradigm with the distributed object model of Java RMI, providing a platform which is suitable for developing partitionable distributed applications. Jgroup depends on RMI in all its interactions; whether internal for coordination between object group replicas, or external for communicating clients with object group. Because of the dynamic of network which is caused by joining new servers and leaving another ones to object group, or caused by partitioning, Partitionable Group Membership Service tracks this changes to provide each member with a report called view. The view contains a list of members which can communicate and coordinate activities.

The advantage of group membership in Jgroup is the ability to continue in providing service in each partition, instead of limiting it in one partition.

When partitions merge, State Merging Service of Jgroup constructs a new global consistent state, to reconcile any divergence caused by conflict updates in the different partitions.

Group Membership Service is required that a view is installed only after agreement is reached on its composition among the servers included in the view (Agreement On View property). To achieve this property; many of Estimation messages are exchanged between the servers, which causes overhead on the network.

This article improves the performance of group membership algorithm which is responsible for achieving the agreement, through allowing for the first server detects the new change in membership to send its estimation to other servers, instead of doing that by each server.

Results show that the enhanced algorithm reduces the number of exchanged estimate messages, and takes approximately the same period of time to reach to agreement on view as in the default algorithm.

Keywords: Object Group; View-oriented Group Communication Systems; Jgroup; Group Membership Algorithm.

[*] **Professor, Department of Systems and Computer Networks, Faculty of Information Engineering, Tishreen University, Lattakia, Syria.**
[**] **Lecturer, Department of Systems and Computer Networks, Faculty of Information Engineering, Tishreen University, Lattakia, Syria.**
[***] **Postgraduate Student, Department of Systems and Computer Networks, Faculty of Information Engineering, Tishreen University, Lattakia, Syria.**

مقدمة:

أدّى افتقار منصّات عمل الوسيط البرمجي (Middleware Framework) مثل RMI، و CORBA [10] لنموذج التفاعل one-to-many إلى عدم دعمها لشفافية التضاعف (Replication Transparency)، والتي تعتبر هامّة لتحقيق التوافرية في الأنظمة الموزعة. ساعدت أنظمة اتصالات المجموعة الموجّهة بالمنظار (view-oriented Group communication system) [7] مثل JavaGroups [8] على تحقيق توافرية الخدمة؛ من خلال إنشاء نسخ متعددة من أغراض المخدم (Replicas) وتوزيعها جغرافياً على عدة أجهزة.

تتميز Jgroup [1] عن غيرها من أنظمة اتصالات المجموعة بدمجها لنموذج مجموعة الغرض [9] مع نموذج الغرض الموزّع RMI، فقدّمت بذلك نظاماً متكاملاً يعتمد تقنية واحدة (RMI) في جميع تفاعلاته، سواء الداخلية لتحقيق التنسيق بين نسخ الخدمة أو الخارجية اللازمة لتحقيق اتصال الزبون مع مجموعة المخدم.

طُوّرت Jgroup إلى منصّة العمل Jgroup/ARM (Autonomous Replication Management)[6]، والتي تسمح بإنشاء نسخ المخدّم على الأجهزة آلياً دون تدخل بشري، وتحافظ على عدد محدد وثابت من النسخ في بيئة الهدف من خلال مراقبتها عبر مكوّن مدير النسخ (Replication Manager). طوّر ميلينغ المنصة ARM إلى منصة جديدة Jgroup/DARM (Distributed Autonomous Replication Management) [5]، يتم فيها توزيع النسخ بطريقة موزّعة دون الحاجة إلى المكوّن المركزي المتمثّل بمدير النسخ المستخدم في ARM.

اقترح موقع مشروع Jgroup [1] إضافة وتحقيق مزايا جديدة في المنصة، كدمج قاعدة بيانات مع Jgroup باستخدام Hibernate، وتحقيق خدمة JavaSpaces مكررة باستخدام Jgroup، تحقّق هذه المقالة ميزة أخرى تم طرحها؛ وهي تحسين خوارزمية عضوية المجموعة المستخدمة في Jgroup.

تشكّل المجموعة الحالية من المخدمات والتي تعتبر أعضاءاً من مجموعة الغرض منظار المجموعة (group view). يمكن أن تتغير عضوية المجموعة عبر الزمن [7]، كانضمام مخدمات جديدة أو مغادرة بعض المخدمات للمجموعة، أو كمغادرة مخدم للمجموعة نتيجةً لتعطّله. تضاف المخدمات وتحذف من منظار المجموعة من خلال أحداث تغيّر المنظار(view change) التي تعالجها خدمة عضوية المجموعة وترسلها إلى نسخة المخدم لتقوم بتحميل منظار جديد. حتى يتمكن كل مخدم من تحميل منظار يحوي تلك المخدمات الممكن الوصول إليها، ويستثني من منظاره تلك المخدمات التي لا يمكن الوصول إليها، تم وضع متطلّب مكوّن من خمس خصائص [7] يحفظ علاقات الوصولية وعدم الوصولية بين المخدمات بشكل دائم وهي:

a) دقّة المنظار (View Accuracy): إذا كان العضو q صحيحاً وقابلاً للوصول من مخدم صحيح آخر p، فإنّ المنظار الحالي للعضو p سوف يحوي دوماً في النهاية العضو q.

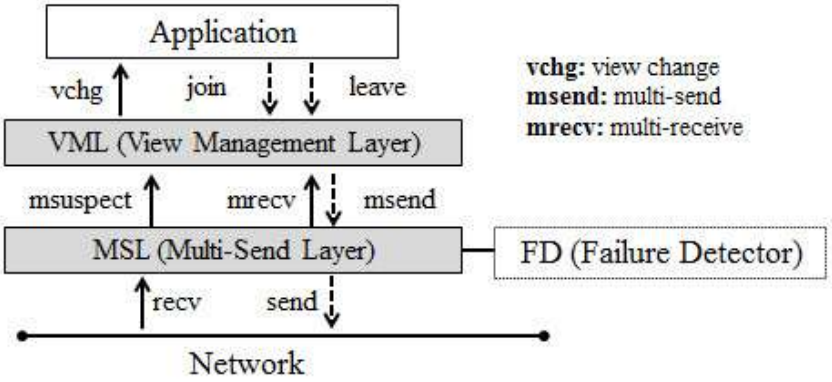
b) اكتمال المنظار (View Completeness): عندما يكون العضو q قد فشل (failed)، أو غادر المجموعة بشكل طوعي، فإنّ المنظار الحالي لكلّ عضو صحيح p سوف يستثني دوماً العضو q.

c) ترابط المنظار (View Coherency): عندما ينصّب العضو الصحيح p منظاراً v، فإنّه؛ ومن أجل كلّ عضو q في المنظار v، إما يُنصّب المنظار v في النهاية، أو أن يكون q غير صحيح وبالتالي سوف ينصّب p المنظار التالي للمنظار v وليكن v1 والذي يستثني العضو q.

d) ترتيب المنظار (View Order) : يجب على المخدمات القابلة للوصول فيما بينها أن تحمّل المناظير نفسها وبالترتيب نفسه.

e) تكامل المنظار (View Integrity): كلّ منظار منصّب بواسطة عضو p سواء أكان صحيحاً أم لا، يحوي p نفسها.

ينبغي على خدمة العضوية في Jgroup تلبية هذه الخصائص الخمس. تم تحقيق الخوارزمية التي تنفّذها خدمة العضوية في [12]، حيث قامت بتوصيف البنية العامة لخدمة عضوية المجموعة (الشكل 1). تتكون خدمة العضوية من طبقتين هما: طبقة البثّ المتعدّد للرسائل وطبقة إدارة المنظار. يزوّد مكتشف الفشل (FD) مجموعات الوصولية إلى طبقة إدارة المنظار (VML) من خلال إرسال الحدث msuspect عبر طبقة البث المتعدّد. تشكّل مجموعة الوصولية المستلمة أساساً للمنظار المراد تشكليه، ولكن يتوجب في البداية التوافق على هذا المنظار قبل تحميله على باقي أعضاء هذا المنظار(الخاصية c). يتم تحميل المنظار الجديد بعد حصول التوافق من خلال إرسال حدث تغيّر المنظار (vchg) من طبقة إدارة المنظار إلى طبقة التطبيق. تخفّض هذه المقالة درجة تعقيد خوارزمية العضوية المطلوبة لتحقيق خاصية ترابط المنظار من $O(N^2)$ إلى $O(N)$ من خلال التقليل من عدد الرسائل المتبادلة بين المخدمات لتحقيق ذلك، مع المحافظة على سرعة الوصول إلى هذا التوافق.



الشكل 1: البنية العامة لخدمة عضوية المجموعة القابلة للتجزئة في Jgroup.

يتم توضيح خطوات عمل الخوارزمية في الفقرة 5 من هذه المقالة لعرض التعديلات التي تم إدخالها لتحسين أدائها، بينما نعرض فيما يلي عدداً من المساهمات التي تم تقديمها لحل مشكلة عضوية المجموعة في أنظمة اتصالات المجموعة. تم تقديم مساهمة لحلّ مشكلة عضوية المجموعة في حالة النظام غير متزامن مع وجود حالات تعطّل (crash) [13]. تحقّق الخوارزمية المقدّمة متطلّباً يسمى بالتاريخ المتّسق requirement (consistent history)، حيث تتوافق المخدمات على ترتيب (sequence) المناظير المحمّلة.

تم تصميم خدمة عضوية المجموعة Moshe [14] للاستخدام في شبكات WANs (Wide Area Networks). تتميّز هذه الخوارزمية بعدم حاجتها للاحتفاظ بمعلومات العضوية من قبل كل مخدّم، حيث يتم تحديد مخدّمات مخصّصة مسؤولة عن تزويد معلومات العضوية تسمى مخدمات العضوية. تجعل هذه الميزة من الخوارزمية قابلة للتوسع وتتجنّب الغمر (flooding) عبر الشبكة، حيث تسمح بنشر معلومات العضوية عند الحاجة لها فقط. تتطلّب Moshe دورات تبادل رسائل متعددة بين الأعضاء قبل التوصّل إلى توافق على المنظار النهائي، لذلك تم اقتراح سلوك تطبيقي جديد في خدمة العضوية [15]، يحقّق التوصّل إلى توافق بين الأعضاء على المنظار النهائي من خلال رسالة تأخير واحدة فقط (رسالة بثّ متعدّد واحدة من كل مخدّم عضوية). تستخدم الخوارزمية تقنية مشابهة للفلترة للتقليل من عدد المناظير المتناقضة قبل الوصول إلى منظار نهائي متوافق عليه.

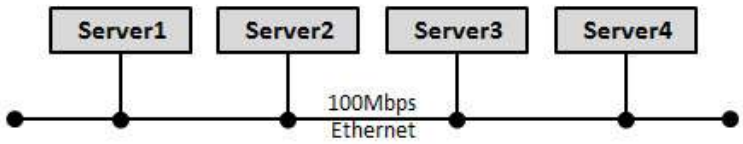
بعد توضيح أهمية البحث والمنهجية المتبعة لتحقيق أهدافه، تتحدث الفقرة4 من هذه المقالة عن خدمة عضوية المجموعة في منصة عمل مجموعة الغرض الموزع Jgroup. توضّح الفقرة 5 خطوات عمل خوارزمية عضوية المجموعة في Jgroup بشيء من التفصيل مع توضيح التعديلات المدخلة عليها بهدف تحسين أدائها. تبين الفقرة 6 مناقشة للنتائج التي تم التوصل إليها بعد إجراء التعديلات. تختتم الفقرة 7 هذه المقالة من خلال توضيح مجموعة من التوصيات والأعمال المستقبلية المقترحة.

أهمية البحث وأهدافه:

تعتبر خدمة عضوية المجموعة الخدمة الأساسية في أنظمة اتصالات المجموعة، وتتميّز هذه الخدمة في Jgroup بأنها قابلة للتجزئة، فهي تسمح بتوافرية الخدمة في جميع أجزاء الشبكة على الرغم من حدوث انقطاع في شبكة الاتصال. لتحقيق متطلّب ترابط المنظار، تبيّن أن خوارزمية العضوية التي تعتمدها الخدمة تتطلب مبادلة عدد من رسائل التخمين بين المخدمات. يهدف هذا البحث إلى تحسين خوارزمية العضوية من خلال التقليل من عدد هذه الرسائل مع المحافظة على الزمن اللازم للوصول إلى حالة التوافق.

منهجية البحث:

تمّ تحضير بيئة الهدف في مخبر الدراسات العليا في كلية الهندسة المعلوماتية بجامعة تشرين. تتألف هذه البيئة الموضّحة في الشكل(2) من أربعة أجهزة متصلة عبر شبكة محلية (Ethernet 100Mbps)، حيث يتم على كل جهاز منها تشغيل مخدّم Jgroup.



الشكل 2: بيئة الهدف.

يتطلّب تشغيل تطبيق Jgroup تحميل البرمجيات الموضّحة في الجدول (1) وذلك على كل جهاز في بيئة الهدف.

جدول1 : البرمجيات المطلوبة لتشغيل تطبيقات Jgroup

Jgroup 3.0.1 distribution [1]
Java J2SE version 1.8.0-20 [2]
Apache Ant version 1.8.4 [3]
Apache Log4j version 1.3-alpha-8 [4]

طوّرت Jgroup باستخدامJAVA J2SE version1.5، ويمكن تحميل هذه النسخة أو أية نسخة أحدث منها (هنا تم اختيار النسخة الأحدث 1.8). تزوّد الحزمة Apache Ant بنسختها 1.8.4 وسيلة سهلة وبسيطة لترجمة واختبار وتشغيل تطبيقات Jgroup. بينما تستخدم الأداة log4j بهدف تصحيح الأخطاء التي يمكن مواجهتها خلال تشغيل التطبيقات. بعد تحميل هذه الأدوات على الأجهزة في بيئة الهدف؛ يصبح بإمكاننا تشغيل تطبيق Jgroup، وإجراء تعديلات على الحزم البرمجية المسؤولة عن خوارزمية عضوية المجموعة بهدف تحسينها. بعد إدخال هذه

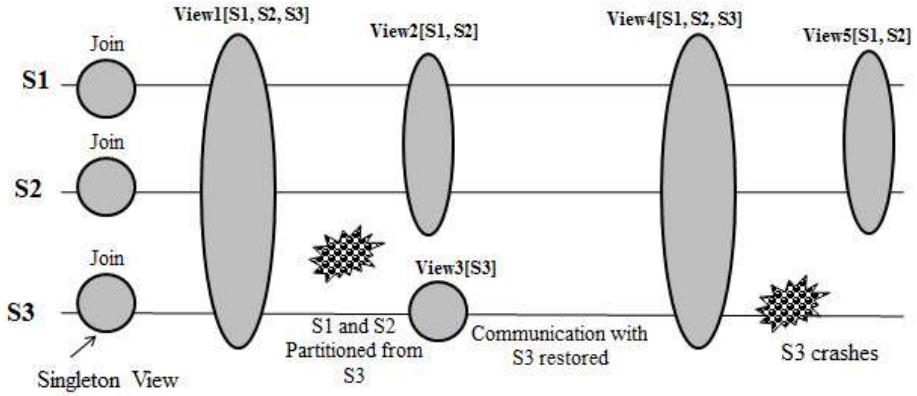
التعديلات؛ تتم المقارنة بين الخوارزمية الأساسية والخوارزمية المحسّنة من حيث عدد رسائل التخمين المتبادلة والزمن اللازم للوصول إلى التوافق على المنظار النهائي؛ وذلك بعد إحداث تغيّر في العضوية كتعطّل أحد المخدمات كالمخدم Server4 مثلاً.

خدمة عضوية المجموعة في منصّة عمل مجموعة الغرض الموزّع Jgroup:

تنسّق أغراض المخدم المشكِّلة لمجموعة الغرض في Jgroup فيما بينها لتزويد خدمات شبكية موزّعة. يزيد استخدام المجموعة من ديناميكية الشبكة؛ من خلال السماح لمخدمات جديدة بالانضمام إلى المجموعة ومغادرتها.

تتبّع خدمة عضوية المجموعة مسار التغيّرات الإرادية الناتجة عن الانضمام والمغادرة؛ بالإضافة إلى التغيرات غير الإرادية الناتجة عن حالات تعطّل إحدى المخدمات أو انقطاع في شبكة الاتصال، فهي تزوّد كل مخدم بمنظار متوافق مع العضوية الحالية في المجموعة. يتألف المنظار من قائمة عضوية مع معرّف فريد؛ ويكون المنظار متوافق مع التركيب الحالي للمجموعة كما تمت معرفته من قبل الأعضاء في هذا المنظار.

يوضح الشكل3 كيفية عمل خدمة عضوية المجموعة، عند حدوث التجزئة بسبب انقطاع الاتصال عبر الشبكة، وعند حدوث حالة تعطّل إحدى المخدمات. يقوم كلّ مخدم بعد انضمامه إلى المجموعة بتشكيل منظار مفرد (singleton view)، حيث تشكّل خدمة العضوية المنظار View1= $[S1, S2, S3]$ الذي يتم تحميله على كل مخدم . ينفصل بعد ذلك المخدمان S1 and S2 عن المخدم S3 بسبب انقطاع في شبكة الاتصال. تستجيب خدمة العضوية عن طريق تحميل منظارين جديدين وهما: View3 =[S3] and View2= $[S1, S2]$. لاحقاً، تختفي التجزئة بعد عودة الاتصال، ويصبح بإمكان المخدمات أن تكوّن منظاراً جديداً View4= $[S1, S2, S3]$ محتوياً جميع المخدمات في المجموعة. تكتشف بعد ذلك خدمة العضوية تعطل المخدم S3 لتقوم بتشكيل منظار جديد View5= $[S1, S2]$.



الشكل 3: كيفية عمل خدمة عضوية المجموعة في Jgroup.

توصيف خوارزمية عضوية المجموعة المحسّنة في Jgroup:

تحقّق هذه الخوارزمية خاصية ترابط المنظار، ويتم تنفيذها ضمن طبقة إدارة المنظار (الشكل 1). تزوّد طبقة الإرسال المتعدد طبقة إدارة المنظار مع مجموعة الوصولية التي تشكّل أساساً للمنظار المراد تحميله. يتمثّل الاختلاف الأساسي بين مجموعة الوصولية والمنظار المراد تشكيله في خاصية الترابط (coherency)، حيث أن مجموعة الوصولية تكون مستقّلة ويمكن تشكيلها دون اتصالات مع باقي أعضاء المجموعة، أما المنظار المراد تشكيله فإنه يتطلّب إرسال رسائل وصولاً إلى التوافق بين جميع الأعضاء ضمنه على تركيب المنظار الجديد.

يمر المخدم خلال عمل الخوارزمية **بمرحلتين** هما: مرحلة الخمول (idle phase) ومرحلة التوافق (agreement phase). تنقسم مرحلة التوافق بدورها إلى مرحلتين فرعيتين هما: مرحلة التزامن Synchronization-phase ومرحلة تبادل التخمين Estimate-Exchange-phase. توضّح الفقرات التالية الخطوات التي يجريها المخدم ضمن كل مرحلة وكيفية انتقاله من مرحلة إلى أخرى، مع توضيح للتعديلات المدخلة على كل مرحلة.

5-1- مرحلة الخمول (Idle): يتواجد المخدم في البداية ضمن هذه المرحلة، حيث يهيّئ ضمنها مجموعة من المتغيّرات (الشكل 4)، من هذه المتغيّرات مصفوفة الأرقام التعاقبية version المستخدمة للتمييز بين مراحل التوافق المتتالية، والمتغيّر synchronized الذي يحوي المخدّمات التي تعلم الرقم التعاقبي الحالي للمخدم p. يمثّل المتغير reachable مجموعة الوصولية للمخدم كما تم استلامها من طبقة البثّ المتعدّد. يحمّل المخدم بداية منظاراً يتكوّن من المخدم نفسه فقط، وذلك بشكل مستقل؛ دون الحاجة لإجراء توافق مع باقي المخدّمات على هذا المنظار. يدوم هذا المنظار حتى وقوع أول استدعاء (msuspect) يشير إلى استلام مجموعة وصولية جديدة. يخرج المخدم p من مرحلة الخمول ليدخل مرحلة التزامن في إحدى الحالتين التاليتين:

- ✓ استلام **حدث تغيّر وصولية** (السطر 13 من الشكل 4) من طبقة البثّ المتعدّد من خلال تنفيذ الاستدعاء msuspect، يسبب ذلك تعديل المتغير reachable الذي يمثل مجموعة وصولية جديدة.
- ✓ استلام **رسالة تزامن** (SYNCHRONIZE) من مخدّم آخر دخل مسبقاً في مرحلة التزامن (السطر 20 من الشكل 4). تحوي رسالة التزامن رقماً تعاقبياً لمرحلة التزامن، وذلك للتمييز بين الاستدعاءات المتتالية لمراحل التوافق على المنظار. قبل دخول المخدّم في مرحلة التزامن، يحدّث مداخل المصفوفة Symset لكلّ المخدّمات التي أصبحت قابلة للوصول بعد آخر حدث msuspect، ويرسل رسالة SYMMETRY إليهم. تجبر الرسالة SYMMETRY المخدم q المرسلة إليه على حذف المرسل p من تخمين منظاره وتسمح لـ q بالوصول إلى توافق على منظار جديد دون انتظار p لتحقيق التوافق معه. إضافة لذلك، تحمل الرسالة SYMMETRY مجموعة المخدمات الموافقة لقيمة المتغير reachable من المرسل، وهذه المجموعة ضرورية لتلبية خاصية اكتمال المنظار، فعندما يستلم مخدم رسالة SYMMETRY؛ يستثني من تخمين منظاره جميع المخدمات القابلة للوصول فعلياً من المرسل. تخزّن القيمة reachable ضمن المصفوفة Symset لتستخدم بطريقة مشابهة لرسائل SYNCHRONIZE، ويحدّث p بعدها المتغيّر reachable. يستدعي المخدم في النهاية التابع ()synchronizationPhaseStart ليبدأ مرحلة التزامن (السطران 19 و25 من الشكل4).

تتمثل **التعديلات المدخلة** إلى مرحلة الخمول والموضّحة على الشكل 4 بإضافة المتغير البولياني increaseVersion وتمريره إلى التابع ()synchronizationPhaseStart، وإضافة المتحول max_Version لتخزين قيمة أكبر رقم تعاقبي. يمكن توضيح الهدف من ذلك على النحو التالي: **يحسّن السلوك الجديد أداء الخوارزمية من خلال تقليل عدد رسائل التخمين المتبادلة بين المخدمات**، فهو يسمح لمخدم واحد فقط بإرسال تخمينه؛ وهو أول مخدم دخل مرحلة التزامن نتيجة حدث msuspect. لتمييز هذا المخدم عن غيره، نسمح له بزيادة رقمه التعاقبي ضمن المصفوفة version عند دخوله لمرحلة التزامن بقيمة أكبر من باقي المخدمات، وذلك من خلال إعطاء المتغير increaseVersion القيمة true (السطر 18 من الشكل4) وتمرير هذه القيمة إلى تابع تهيئة مرحلة التزامن،

في حين تزيد باقي المخدمات التي تدخل مرحلة التزامن نتيجة استلامها لرسالة تزامن من مخدم آخر رقمها التعاقبي بقيمة أقل (1 فقط)، وتكون قيمة increaseVersion تساوي false (السطر 24 من الشكل4).

```
1   class JgroupDaemon {
2     JgroupDaemon(Member member) {     /* Constructor */
3       reachable = {p}                 /* Set of unsuspected objects */
4       version = (0, . . . , 0)        /* Vector clock */
5       Boolean increaseVersion = false;
6       max_Version = version[p];
7       symset = ({p}, . . . , {p})     /* Symmetry set */
8      view = ( UniqueID( ), {p} )      /* Current view id and composition */
9      cview = view                     /* Corresponding complete view */
10     this.member = member;    /* Member reference used to notify upcalls */
11     member.vchg(view);
12       } /* End Constructor */
13    void msuspect (MemberId[] P): // received from MSL
14    foreach (r ∈ ( Π – P) – reachable) do {
15    symset[r] = reachable ;
16    msend( ⟨SYMMETRY, version, reachable⟩, (Π – P) – reachable);
17     reachable = Π – P ;
18    increaseVersion = true ; /* to increase version number in s-phase */
19     synchronizationPhaseStart(increaseVersion); }
20    void mrecv (⟨SYNCHRONIZE, Vp, Vq, P⟩, MemberId q) {
21    if (version[q] < V [q]) {
22      version[q] ← V[q]    // modify version array for sender (q)
23    if (q ∈ reachable)
24      increaseVersion = false;
25     synchronizationPhaseStart(increaseVersion); }
26     }
```

الشكل 4: مرحلة الخمول idle للمخدم P.

5-2- مرحلة التزامن: يتم في هذه المرحلة تبادل رسائل تزامن تتوافق فيها جميع المخدمات على مداخل المصفوفة version التي تحوي أرقاماً تعاقبية من جميع المخدمات، حيث تميّز هذه المصفوفة بين مراحل التوافق، وكل تغيير في إحدى مداخل هذه المصفوفة يشير إلى الدخول في مرحلة توافق تالية نتيجة حدوث تغير في العضوية. يهيئ التابع ()synchronizationPhaseStart هذه المرحلة (الشكل5) ويبدأ المخدم بتخمين المنظار التالي. في حال دخل p مرحلة التزامن نظراً لاستلامه رسالة تزامن m من مخدّم q، فإنّ رسالة الإجابة المرسلة من p تمثّل ردّاً على m.

تدوم مرحلة التزامن على مخدم p حتى تجيب جميع المخدّمات في تخمين المنظار على رسالة التزامن من p (يتم اختبار ذلك من خلال التابع ()checkSynchronization السطر 43 من الشكل 5)، أو عندما يستلم p رسالة تخمين من مخدم آخر قد دخل مسبقاً في مرحلة تبادل التخمين (السطر 33 من الشكل5). يدخل المخدّم مرحلة تبادل التخمين e-phase من الخوارزمية في حال كان المتغيّر *synchronized* يحوي جميع المخدّمات الموجودة ضمن المتغيّر *estimate*، ويتم اختبار هذا الشرط ضمن ()checkSynchronization.

لضمان إنهاء مرحلة التزامن، في كلّ مرة يستثني المخدم p مخدماً آخر q، أو يستلم رسالة SYMMETRY من q، يحذف p المخدّم q من تخمين منظاره، كما أن المخدّم p عند استلامه رسالة SYNCHRONIZE من مخدّم

q يحوي رقمه التعاقبي، فإنّه يضيف q إلى *synchronized*. في هذه الحالة سوف يحتوي المتغيّر *synchronized* في النهاية على جميع المخدمات الموجودة في *estimate* ويتحقق شرط الخروج من هذه المرحلة

```
1  synchronizationPhaseStart(Bool  increaseVerion)  {
2     estimate =  reachable                /* Next view estimation */
3     synchronized = {p}          /* Synchronized servers */
4     if  (increaseVersion)  then
5        version[p] = version[p] + randomInt + 1 ;       /* Generatea  new version number*/
6     else  version[p] = version[p] + 1;
7     max_Version = version[p];
8     foreach r ∈ estimate – {p}  do
9           msend(⟨SYNCHRONIZE,  version[r], version[p], symset[r]⟩, {r})  }
10 void msuspect(MemberId[] P) {
11     foreach r ∈ (Π– P) – reachable)  do symset[r] = reachable;
12     msend(⟨SYMMETRY, version, reachable⟩, (Π– P) – reachable);
13     reachable ← Π– P;
14     estimate = estimate ∩ reachable;
15     send_Estimate =  false;          // optimization   --- this message to exit from s-phase.
16     checkSynchronization(send_Estimate); }
17 void mrecv(⟨SYMMETRY, V, P⟩, MemberId  q) {
18     if (version[p] == V [p]) and (q ∈ estimate) then
19           estimate = estimate – P;
20           send_Estimate =  false;      //  optimization --- this message to exit from s-phase.
21           checkSynchronization(send_Estimate);   }
22 void mrecv(⟨SYNCHRONIZE, Vp, Vq, P⟩, MemberId q)  {
23     if (version[p] == Vp) then {
24                   synchronized = synchronized ∪ {q};      // q  knew my version number
25                   agreed[q] =Vq ; }
26     if (version[q] < Vq) {
27              version[q] =Vq ;
28              msend(⟨SYNCHRONIZE, version[q], version[p], symset[q]⟩; {q}); }    // End  if
29 if ( version[q] > max_Version )     {  max_Version = version[q];     }
30 if (( version[p] = = max_Version )   send_Estimate = true;
31       else send_Estimate = false;    // optimization
32       checkSynchronization(send_Estimate);   }    // end if }
33 void mrecv(⟨ESTIMATE, V,  P⟩, Member q) {
34    version[q] = V [q]
35           if (q ∉ estimate) then
36                msend(⟨SYMMETRY, version, estimate⟩, {q});
37          else if (version[p] = = V [p]) and (p ∈ P) then
38                estimate = estimate ∩ P ;
39                synchronized = P ;
40                agreed = V ;
41               send_Estimate = false; }
42   checkSynchronization(send_Estimate);  }
43 void checkSynchronization(Bool send_Estimate) {
44      if (estimate ⊂ synchronized)      estimateExchangePhaseStart(send_Estimate);  }
```

الشكل 5: مرحلة التزامن للمخدم P بعد إدخال التعديلات.

في حالة ثانية؛ يمكن أن يدخل المخدم p مرحلة تبادل التخمين عند استلامه رسالة تخمين <Estimate, V,P> من مخدّم q، والذي يعلم الرقم التعاقبي الحالي لـ p ولم يحذفه من تخمين منظاره، عند ذلك تعدّل p كل من *synchronized* و*estimate* لتضمن الخروج من مرحلة التزامن وتدخل في مرحلة تبادل التخمين. بعد دخول المخدم

في مرحلة تبادل التخمين يكون قد تم الاستقرار على مصفوفة version ثابتة ومتفق عليها تعبّر عن مرحلة التوافق الحالية، ويتم تخزين قيمها ضمن المصفوفة agreed.

تتمثّل **التعديلات المدخلة على مرحلة التزامن** في الخوارزمية المحسّنة في طريقة تعديل الرقم التعاقبي للمخدمات (الأسطر 4و5و6 من الشكل 5). فالمخدم الذي دخل هذه المرحلة مع القيمة true للمتغير increaseVersion يزيد رقمه التعاقبي بقيمة أكبر من باقي المخدمات بقيمة عشوائية، أما المخدم الذي يدخل هذه المرحلة مع القيمة false فإنه يزيد رقمه التعاقبي بمقدار 1 كما هو الحال في الخوارزمية الأساسية. لقد تم اختيار الزيادة وفقاً لقيمة عشوائية منعاً لتساوي رقمين تعاقبيين لمخدمين يدخلان سوية مرحلة التزامن بسبب حدثي msuspect متزامنين. مع استلام المخدم لرسائل تزامن من باقي المخدمات يخزن أكبر رقم تعاقبي مستلم ضمن max_Version (السطر 29 من الشكل 5). بعد استلام جميع رسائل التزامن، إذا كان الرقم التعاقبي للمخدم هو الأكبر ضمن المصفوفة version فإنه يتوجب عليه إرسال تخمينه بعد دخوله إلى مرحلة تبادل التخمين، عندها يأخذ المتحول البولياني الجديد send_Estimate الممرّر إلى تابع اختبار الخروج من مرحلة التزامن checkSynchronization() القيمة true (السطر 30). عدا ذلك يدخل المخدم مرحلة تبادل التخمين دون الحاجة لإرسال تخمينه **(وهنا يتم التقليل من عدد رسائل التخمين المتبادلة).** يعود سبب منعنا هذا المخدم من إرسال تخمينه؛ إلى أن هذا المخدم سوف يعلم مجموعة الوصولية الجديدة ويعدّل تخمينه وفقاً لتخمين المرسل، وبالتالي لا داعي أن يعيد إرسال هذا التخمين الجديد إلى باقي المخدمات التي يتم إبلاغها بالعضوية الجديدة من قبل المخدم الذي يملك أكبر رقم تعاقبي ضمن version.

5-3- مرحلة تبادل التخمين: يبدأ المخدم هذه المرحلة (الشكل 6) بعد خروجه من مرحلة التزامن باستدعائه التابع estimateExchangePhaseStart(). يتم ضمنه استدعاء الطريقة sendEstimate() لتعديل تخمين المنظار وإعلام باقي المخدّمات عن حدوث التغيير. في نفس الوقت وخلال استدعاء هذه الطريقة، يقوم p بإرسال رسالة مقترح propose تحمل ضمنها حالته الحالية إلى منسّق منتخب من تخمين منظاره. يجعل هذا السلوك المخدم يقوم بإرسال تخمينه مع كل تعديل في هذا التخمين إلى باقي المخدمات. للاستفادة من **التعديلات المدخلة** في مرحلة التزامن على زيادة الأرقام التعاقبية وضبط قيمة المتغير send_Estimate؛ فإن التخمين يتم إرساله عندما يملك هذا المتغير القيمة true، وهي تكافئ حالة المخدم الذي يملك أكبر رقم تعاقبي. أما المخدم الذي يدخل مرحلة تبادل التخمين مع القيمة false للمتغير send_Estimate فإنه لا يرسل تخمينه إلى باقي المخدمات وإنما يقوم فقط بإرسال مقترحه عن المنظار إلى المنسق.

تمرر قيمة send_Estimate إلى تابع تهيئة مرحلة تبادل التخمين (السطر 1 من الشكل 6)، ويلاحظ عملية اختبار المتغير قبل القيام بإرسال التخمين (السطر 5 من الشكل 6).

لا يطرأ أي تعديل على باقي أجزاء الخوارزمية، حيث يُستدعى التابع checkAgreement(C) (السطر 7 من الشكل 6) عند استلام المخدّم (المنسّق) لرسائل المقترحات. حيث يتحقّق فيما إذا كانت المقترحات المخزنة في جدول المنسّق متساوية، من خلال اختبار فيما إذا كانت جميع تخمينات المنظار متساوية وفيما إذا كانت جميع المخدّمات تعلم نفس مصفوفات الرقم التعاقبي (محدودة إلى مخدّمات في تخمين المنظار نفسه).

أخيراً تستخدم الطريقة *installView()* (السطر 10 من الشكل 6) لتحميل منظار جديد. حيث يوجّه أولاً p رسالة منظار VIEW تحتوي معرّف المنظار الجديد وجدول المنسّق C إلى جميع المخدمات التي كانت قد وصلت إلى

التوافق. تعتبر هذه المعلومات ضرورية على اعتبار أنه من الممكن تعطّل المنسّق قبل إرسال المنظار إلى جميع المستلمين.

```
1   void estimateExchangePhaseStart(Bool send_Estimate) {
2         sendEstimate(∅, send_Estimate); }
3   void sendEstimate(MemberId[] P, Bool send_Estimate) {
4      estimate = estimate – P;
5      if (send_Estimate)   msend(⟨ESTIMATE, agreed, estimate⟩, reachable − {p});
6      msend(⟨PROPOSE, (cview, agreed, estimate)⟩, Min(estimate));
    // to coordinator  }
7   boolean checkAgreement(CoordinatorTable C) {
8        return (∀q ∈ C[p].estimate : C[p].estimate = = C[q].estimate)
9        and (∀q, r ∈ C[p].estimate : C[p].agreed[r] = = C[q].agreed[r]);  }
10  void installView(View w, Coordinator C) {
11     msend(⟨VIEW, w, C>, C[p].estimate – {p});
12   if (∃q, r ∈ C[p].estimate : q ∈ C[r].cview.comp and C[q].cview.id ≠
13  C[r].cview.id) then
14  view ← ( (w, view.id), {r | r ∈ C[p].estimate and C[r].cview.id = = cview.id});
15    else   view = ((w,⊥);C[p].estimate);
16       member.vchg(view);
17       cview = (w, C[p].estimate)
18  stable = (view.comp = = reachable) and (∀q, r ∈ C[p].estimate : C[p].agreed[r] = = agreed[r]);
19  if (stable)     idlePhaseStart();
20   else synchronizationPhaseStart();    ……  }
```

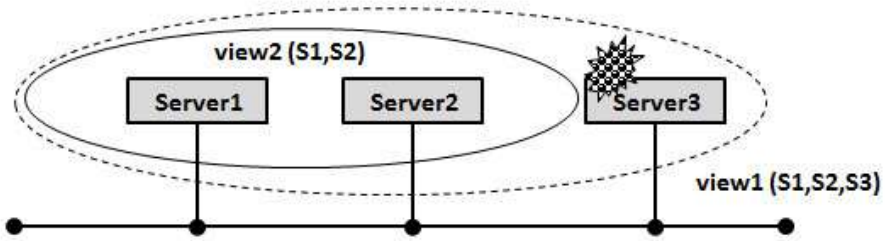
الشكل 6 : مرحلة تبادل التخمين للمخدم p والتعديلات المدخلة عليها.

تتم معالجة الأحداث خلال مرحلة تبادل التخمين في الخوارزمية المحسّنة بنفس الطريقة التي تعالج بها في الخوارزمية الأساسية، ولكن ينبغي فيها تعديل صيغة التابع sendEstimate بحيث يمرر له المتغير البولياني المضاف (send_Estimate).

النتائج والمناقشة:

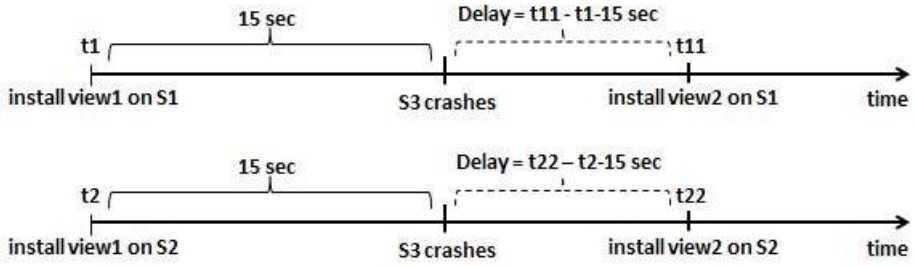
لإظهار فعالية التحسين المدخل على خوارزمية عضوية المجموعة، تم بناء تطبيق Jgroup يسمح بإنشاء مجموعة من أغراض المخدم على الأجهزة الموضحة في بيئة الهدف (الشكل 2). يسمح هذا التطبيق بالمقارنة بين الخوارزمية الحالية والخوارزمية المحسنة من حيث عدد رسائل التخمين المتبادلة والزمن اللازم للوصول إلى التوافق على المنظار؛ وذلك بعد إحداث تغيّر في عضوية المجموعة ناتج عن تعطيل إحدى المخدمات الأعضاء. ونظراً لعدم إمكانية معرفة الزمن اللازم للوصول إلى التوافق، فإنه تم قياس زمن التأخير اللازم لخدمة العضوية حتى تقوم بتحميل المنظار التالي على كل مخدم عضو بعد إحداث تغير في العضوية؛ على اعتبار أن تحميل المنظار مرتبط بالوصول إلى التوافق، حيث لا يمكن أن يتم تحميل هذا المنظار إلا اذا كان قد حصل التوافق عليه بين الأعضاء الموجودة ضمن تركيبه.

1- السيناريو الأول: يوضح الشكل 7 هذا السيناريو، حيث تتكون مجموعة المخدم من ثلاثة مخدمات (S1,S2,S3)، ويتم تعطيل المخدم Server3 بعد فاصل زمني معين.



الشكل 7: سيناريو 1 (تعطّل Server3 وتشكيل المنظار view2 على Server1 و Server2).

عند انضمام المخدم Server3 إلى مجموعة الغرض يتم تشكيل منظار view1 يحوي المخدمات الثلاثة، حيث يتم تحميل هذا المنظار في اللحظة t1 على المخدم Server1 وفي اللحظة t2 على المخدم Server2 (الشكل 8). بعد فاصل زمني معين (وليكن 15sec) يتم تعطيل المخدم Server3. يسبب ذلك حدوث تغير في العضوية يجعل خوارزمية العضوية تنفذ خطواتها للوصول إلى توافق على المنظار الجديد (view2 في الشكل 7). عندها يتم إرسال عدد من رسائل التخمين بين المخدمين Server2 و Server1 والتي يتم حساب عددها. في النهاية يتم تحميل المنظار view2 في اللحظة t11 على المخدم Server1، وفي اللحظة t22 على المخدم Server2.



الشكل 8: زمن التأخير Delay اللازم لتحميل المنظار التالي، بعد تعطيل المخدم Server3.

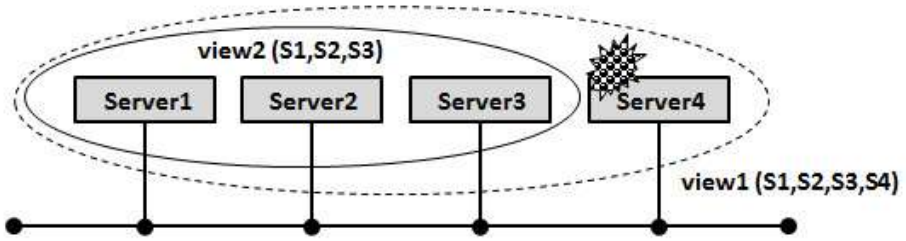
تم حساب التأخير الزمني التالي (t11 – t1 – 15sec) على المخدم Server1 والموضح في الشكل 8، والذي يعبر عن الزمن الذي استغرقته خوارزمية العضوية لتحميل المنظار التالي على Server1 منذ لحظة تعطّل المخدم Server3. وحساب الزمن (t22 – t2 – 15sec) على المخدم Server2، والذي يعبر عن الزمن الذي استغرقته خوارزمية العضوية لتحميل المنظار التالي على Server2 منذ لحظة تعطّل المخدم Server3. ، وذلك في حالتي الخوارزمية الأساسية والخوارزمية المحسنة. تم تكرار تنفيذ التجربة عشر مرات وأخذ المتوسط الحسابي لزمن التأخير. يعرض الجدول (2) النتائج التي تم التوصل إليها في هذا السيناريو.

جدول 2: نتائج السيناريو 1.

Enhanced	Default	
4,984	5,090	Server1(Delay ms)
5,080	5,118	Server2
1	2	Number of Estimates

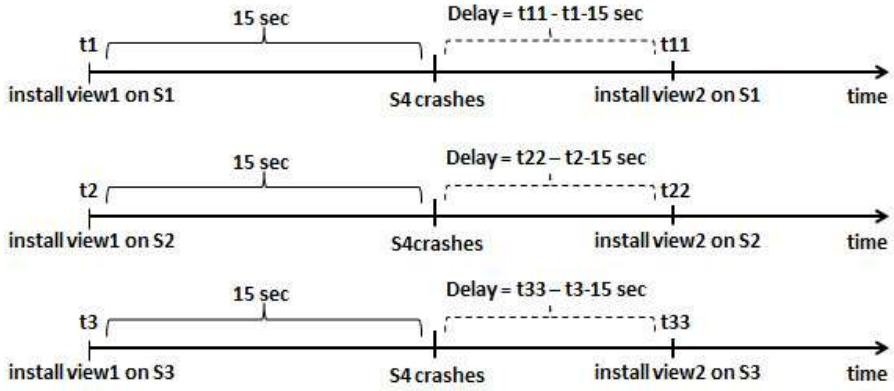
يلاحظ من الجدول 2 انخفاض عدد رسائل التخمين من 2 في الخوارزمية الافتراضية إلى 1 في الخوارزمية المحسنة، نتيجة السماح لمخدم واحد فقط بإرسال تخمينه، في حين يقوم كل مخدم في الخوارزمية الافتراضية بإرسال تخمينه إلى المخدم الآخر، كما يلاحظ أن الخوارزمية المحسنة تستغرق بشكل تقريبي الفترة الزمنية نفسها التي تستغرقها الخوارزمية السابقة لتحميل المنظار (حوالي 5sec).

2- السيناريو الثاني: يوضح الشكل 9 هذا السيناريو، حيث تتكون مجموعة المخدم من أربعة مخدمات (S1,S2,S3,S4)، ويتم تعطيل المخدم Server4 بعد فاصل زمني معين.



الشكل 9: سيناريو2 (تعطّل Server4 وتشكيل المنظار view2 على Server1 وServer2 و Server3).

عند انضمام المخدم Server4 إلى مجموعة الغرض يتم تشكيل منظار view1 يحوي المخدمات الأربعة، حيث يتم تحميل هذا المنظار في اللحظة t1 على المخدم Server1، وفي اللحظة t2 على المخدم Server2، وفي اللحظة t3 على المخدم Server3 (كما هو موضح في الشكل 10). بعد فاصل زمني معين (وليكن 15sec) يتم تعطيل المخدم Server4، مما يحدث تغيراً في عضوية المجموعة يجعل خوارزمية العضوية تنفذ خطواتها للوصول إلى توافق على المنظار الجديد (view2 في الشكل 9). عندها يتم تبادل عدد من رسائل التخمين بين المخدمات Server1 وServer2 وServer3 والتي يتم حساب عددها. في النهاية يتم تحميل المنظار view2 في اللحظة t11 على المخدم Server1، وفي اللحظة t22 على المخدمServer2، وفي اللحظة t33 على المخدم Server3.



الشكل 10: زمن التأخير Delay اللازم لتحميل المنظار التالي، بعد تعطيل المخدم Server4.

يعرض الجدول (3) النتائج التي تم التوصل إليها في هذا السيناريو (زمن التأخير وعدد رسائل التخمين).

جدول 3: نتائج السيناريو2.

Enhanced	Default	
5,539	5,397	Server1(Delay ms)
5,721	5,623	Server2
5,664	5,408	Server3
2	6	Number of Estimates

يلاحظ من الجدول 3 انخفاض عدد رسائل التخمين المرسلة من 6 في الخوارزمية الافتراضية إلى 2 في الخوارزمية المحسنة، ففي الخوارزمية الافتراضية يقوم كل مخدم من المخدمات الثلاثة بإرسال تخمينه إلى باقي المخدمات (6 = 2*3)، في حين يقوم مخدم واحد فقط في الخوارزمية المحسنة بإرسال تخمينه إلى المخدمين الآخرين

(2 فقط). كما تستغرق الخوارزمية المحسنة الفترة الزمنية نفسها بشكل تقريبي التي تستغرقها الخوارزمية السابقة لتحميل المنظار (حوالي 5.5 sec).

الاستنتاجات والتوصيات:

a) يخفّض التحسين المدخل إلى خوارزمية عضوية المجموعة درجة التعقيد من $O(N^2)$ إلى $O(N)$. فإذا افترضنا أن مجموعة غرض المخدم مكوّنة من N مخدم وتعطل أحد هذه المخدمات، فإن عدد رسائل التخمين المتبادلة بعد اكتشاف حالة التعطّل هذه يساوي $(N-2)(N-1)$ أي درجة تعقيد $O(N^2)$. أما في الخوارزمية المحسنة فإن تعطل أحد المخدمات يسبب إرسال مخدم واحد فقط لرسالة التخمين إلى $N-2$ مخدم وهذا يكافئ درجة تعقيد $O(N)$.

b) يزداد زمن التأخير اللازم لتحميل منظار جديد بعد حدوث تغيّر العضوية مع تزايد عدد المخدمات في الخوارزمية الافتراضية. ويزداد هذا الزمن بنفس المقدار تقريباً في الخوارزمية المحسنة، فقد ازداد زمن التأخير من 5sec تقريباً في حال مخدمين إلى 5.5sec تقريباً مع وجود ثلاثة مخدمات.

c) لتحقيق تحسين إضافي في أداء خوارزمية عضوية المجموعة، نقترح الاستغناء عن فكرة وجود المنسّق، من خلال احتواء كل مخدم على جدول مشابه لجدول المنسق يسمى جدول التخمين. يحوي هذا الجدول مدخلاً لكل مخدم من مخدمات المجموعة، وعند استلام المخدم رسالة تخمين من مخدم آخر يقوم بتعديل السجل الخاص به ضمن جدول التخمين. بعد استلام المخدم جميع التخمينات، ينفّذ تابعاً جديداً checkEstimation() يختبر من خلاله تساوي هذه التخمينات ليخرج عند تحقق شرط التوافق من مرحلة تبادل التخمين ويقوم بتحميل منظار جديد. يساعد هذا الاقتراح كل مخدم من مخدمات المجموعة على اختبار الوصول إلى التوافق من خلال متحولاته المحلية فقط، دون الحاجة لوجود منسق، كما يمنع الدخول في مرحلة توافق جديدة في حالة تعطّل المنسّق المنتخب وانتخاب منسّق جديد.

d) تم اقتراح آلية لتحديد متى يشكّل منظار جزءاً أساسياً للمجموعة [16]. يمكن من خلال هذه الآلية لكلّ عضو من المجموعة أن يتحقّق فيما إذا كان ينتمي إلى الجزء الأساسي أم لا بناء على معلومات محلية. تعتمد المنهجية المتبعة أسلوب تزامن المنظار المخصّب (Enriched view synchrony) [16] الذي يسهّل حفظ الحالة المشتركة المعرّفة بالتطبيق في مجموعة المخدم. تم في هذه المنهجية استبدال فكرة المنظار العادي بمفهوم المنظار المخصّب Enriched View أو اختصاراً e-view، حيث تجمّع المخدمات في مناظير فرعية subviews، وتجمّع المناظير الفرعية في مجموعات sv-set، والتي بدورها تتواجد ضمن e-view. يسمح إنشاء طبقة جزء أساسي فوق طبقة خدمة عضوية المجموعة القابلة للتجزئة بعزل عملية تطوير التطبيقات المعنية بالتجزئة، بدلاً من تضمين مفهوم الجزء الأساسي ضمن طبقة عضوية المجموعة القابلة للتجزئة. تصبح التطبيقات بعد ذلك قادرة على تزويد طاقم كامل من عملياتها في الجزء الأول فقط (primary-partition)، بينما يمكن لمجموعة فرعية من عملياتها أن تكون متوفرة في الأجزاء غير الرئيسية. إن إنشاء هذه الطبقة في Jgroup يجعلها ملائمة للتطبيقات التي تعتمد مفهوم الجزء الأساسي بالإضافة إلى دعمها الافتراضي للتطبيقات المعنية بالتجزئة. وهو من الأعمال المستقبلية المقترحة أيضاً.

المـراجـع :

[1] UNIVERSITY OF BOLOGNA , 2008, Jan.2017 <http://jgroup.sourceforge.net/index.html>.

[2] SOFTONIC , 2009, 7June.2016. <http://java-development-kit-jdk.en.softonic.com/>.

[3] APATCHE ANT ,2014, 3February.2016 . <http://ant.apache.org/>.

[4] APACHE ,2012, 10July.2016. <http://logging.apache.org/log4j/1.2/>.

[5] MELING, H. *'An Architecture for Self-healing Autonomous Object Groups'.* University of Stavenger, Department of Electrical Engineering and Computer Science, N-4036 Stavenger, Norway, 2008.

[6] MELING, H.; MONTRESOR, A.; HELVIK, B. E. and BABAOGLU, O. *'Jgroup/ARM: a distributed object group platform with autonomous replication management'*, *Softw. Pract. Exper.*, 38: 885–923. DOI: 10.1002/spe.853, 2008.

[7] VITENBERG, R.; KEIDAR, I.; CHOCKLER, G. and DOLEV, D. *'Group Communication Specifications: A Comprehensive Study'*. Technical Report CS99-31, Institute of Computer Science, The Hebrew Univ. of Jerusalem, 1999.

[8] BAN, B. *'JavaGroups:Group communication patterns in Java'.* Technical Report, Department of Computer Science, Cornell University, July 1998.

[9] MONTRESOR, A. *'System Support for programming object-oriented dependable applications in partitionable systems'.* (PhD Thesis), University of Bologna, Department of Computer Science. pp. 87-100. February2000,

[10] DEITEL, H.M., DEITEL, P.J. and SANTY, S.E. *'Advanced Java 2 Platform: How To PROGRAM',* New Tersey: Prentice-Hall, Inc, 2001.

[11] BABAOGLU, O.; BARTOLI, A. 'Selecting a Primary Partition in Partitionable Asynchronous Distributed Systems'. TRANSACTIONS ON COMPUTERS, 1997.

[12] BABAOGLU, O., DAVOLI, R., MONTRESOR, A. *'Group Communication in Partitionable Systems: Specification and Algorithm'*. Tech. Rep. UBLCS95-18, Dept. of Computer Science, University of Bologna, Sep. 1996. To appear in Operating Systems Review, Jan. 1997.

[13] FRANCESCHETTI, M. and BRUCK, J., *'A Group Membership Algorithm with a Practical Specification'***,** IEEE Transactions on Parallel and Distributed Systems, IEEE Press Piscataway, NJ, USA, Volume 12, Page 1190-1200, 11 November 2001.

[14] KEIDAR, I., SUSSMAN, J., DOLEV, D. *'Moshe: A group membership service for WANs'*, ACM Transactions on Computer Systems, 20(3):1 – 48, August 2002.

[15] KHAZAN, I. *'Group Membership: A Novel Approach and the First Single-Round Algorithm'*, ACM Transactions on Computer Systems, 1-58113-802-4/04/0007, July 2004.

[16] BABAOGLU, O.; BARTOLI, A.; and DINI, G. *'Enriched View Synchrony: A Programming Paradigm for Partitionable Asynchronous Distributed Systems'*. IEEE TRANSACTIONS ON COMPUTERS, VOL. 46, NO. 6, JUNE 1997.