14 de febrero de 2020.
Dictamen C.I. 01/2020

DICTAMEN
QUE PRESENTA LA COMISIÓN DE INVESTIGACIÓN DE LA DIVISIÓN DE CIENCIAS DE LA COMUNICACIÓN Y DISEÑO

ANTECEDENTES

I. El Consejo Divisional de Ciencias de la Comunicación y Diseño, en la sesión 10.19, celebrada el 16 de julio de 2019, integró esta Comisión en los términos señalados en el artículo 55 de Reglamento Interno de los Órganos Colegiados Académicos.

II. El Consejo Divisional designó para esta Comisión a los siguientes integrantes:

a) Órganos personales:
- ✓ Dr. Jesús Octavio Elizondo Martínez, Jefe del Departamento de Ciencias de la Comunicación;
- ✓ Mtro. Alejandro Rodea Chávez, Encargado del Departamento de Teoría y Procesos del Diseño;
- ✓ Dr. Carlos Joel Rivero Moreno, Jefe del Departamento de Tecnologías de la Información.

b) Representantes propietarios:
- Personal académico:
  - ✓ Dr. André Moise Dorcé Ramos, Departamento de Ciencias de la Comunicación;
  - ✓ Dra. Deyanira Bedolla Pereda, Departamento de Teoría y Procesos del Diseño.
  - ✓ Dr. Tiburcio Moreno Olivos, Departamento de Tecnologías de la Información.

CONSIDERACIONES

I. La Comisión recibió, para análisis y discusión, el informe final del proyecto de investigación denominado ***"Análisis de arquitecturas colaborativas multi-fuente usando codificación de red"*** presentado por el Dr. Francisco de Asís López Fuentes, aprobado en la Sesión 06.16 celebrada el 6 de junio de 2016, mediante el acuerdo DCCD.CD.08.06.16.

División Ciencias de la Comunicación y Diseño

Unidad Cuajimalpa
DCCD | División de Ciencias de la Comunicación y Diseño
Torre III, 5to. piso. Avenida Vasco de Quiroga 4871,
Colonia Santa Fe Cuajimalpa, Alcaldía Cuajimalpa de Morelos,
Tel. +52 (55) 5814-6553. C.P. 05348, México, D.F.
http://dccd.cua.uam.mx

II. La Comisión de Investigación sesionó los días 13 de enero y 14 de febrero de 2020, fecha en la que concluyó su trabajo de análisis y evaluación del informe final, con el presente Dictamen.

III. La Comisión tomó en consideración los siguientes elementos:

- *"Lineamientos para la creación de grupos de investigación y la presentación, seguimiento y evaluación de proyectos de investigación"* aprobados en la Sesión 06.16 del Consejo Divisional de Ciencias de la Comunicación y Diseño, celebrada el 6 de junio de 2016, mediante al acuerdo DCCD.CD.15.06.16.
- Protocolo de investigación.
- Relevancia para el Departamento.
- Objetivos planteados.
- Resultados obtenidos.

IV. **Objetivos planteados en el proyecto**:

General:

Investigar y analizar una arquitectura multi-fuente que permita integrar técnicas de codificación de red y optimizar la difusión de contenidos en ambientes de redes P2P.

Particulares:

- Estudiar las técnicas de codificación usadas para una arquitectura de red P2P básica.
- Estudiar las técnicas de codificación usadas para una arquitectura de red P2P con múltiples fuentes y múltiples receptores.
- Modelar el comportamiento de redes P2P con múltiples fuentes usando técnicas de codificación de red.
- Estudiar y analizar los datos obtenidos para generar publicaciones nacionales e internacionales.

V. **Resultados alcanzados:**

Se cumplieron los objetivos planteados y se alcanzaron los siguientes resultados:

1. Redes de investigación:
   Se generó un fuerte trabajo colaborativo con investigadores de diferentes centros de investigación a través de la red CONACYT Red en Sistemas y Redes de Próxima Generación, así como con alumnos interesados en la codificación de red.

2. Modelo matemático:
   Se propuso un modelo matemático para una arquitectura basada en árboles P2P altamente colaborativa y escalable hecha de dos componentes principales: 1) Agrupación o agrupación por pares y 2) Distribución de contenido jerárquico basado en árboles.

3. Estancia de investigación corta:
   Se realizaron dos estancias de investigación corta en el CIMAT-Guanajuato durante el mes de agosto de 2017 y agosto de 2018.

4. Formación de recursos humanos:
   El proyecto permitió involucrar a alumnos de la LTSI con un proyecto terminal concluido y un servicio social.

5. Artículos de investigación:
   2 artículos en conferencias internacionales IEEE, 1 artículo en conferencia internacional EAI, 1 artículo en revista con índice LATINDEX, 1 artículo en revista con índice JCR.

DICTAMEN

ÚNICO:

Tras evaluar el informe final del proyecto de investigación *denominado* ***""Análisis de arquitecturas colaborativas multi-fuente usando codificación de red"*** presentado por el Dr. Francisco de Asís López Fuentes, la Comisión de Investigación recomienda al Consejo Divisional de Ciencias de la Comunicación y Diseño aceptarlo.

VOTOS:

Integrantes	Sentido de los votos
Dr. Jesús Octavio Elizondo Martínez	A favor
Mtro. Alejandro Rodea Chávez	A favor
Dr. Carlos Joel Rivero Moreno	A favor
Dr. André Moise Dorcé Ramos	A favor
Dra. Deyanira Bedolla Pereda	A favor
Dr. Tiburcio Moreno Olivos	------
Total de los votos	**5 votos a favor**

Coordinadora

Dra. Gloria Angélica Martínez De la Peña
Secretaria del Consejo Divisional de
Ciencias de la Comunicación y Diseño

Casa abierta al tiempo
UNIVERSIDAD AUTONOMA METROPOLITANA
Unidad Cuajimalpa

Oficina Técnica del
Consejo Divisional

HOJA DE FIRMAS DEL DICTAMEN C.I. 01/2020 DE FECHA 14 DE FEBRERO DE 2020 QUE EMITE LA COMISIÓN DE INVESTIGACIÓN DE LA DCCD.

División Ciencias de la Comunicación y Diseño

Unidad Cuajimalpa
DCCD | División de Ciencias de la Comunicación y Diseño
Torre III, 5to. piso. Avenida Vasco de Quiroga 4871,
Colonia Santa Fe Cuajimalpa, Alcaldía Cuajimalpa de Morelos,
Tel. +52 (55) 5814-6553. C.P. 05348, México, D.F.
http://dccd.cua.uam.mx

ACUSE

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Cuajimalpa

Comunidad académica comprometida con el desarrollo humano de la sociedad.

DCCD.DTI.004.20
Enero 16, 2020

Dr. Octavio Mercado González
Presidente del Consejo Divisional
División de Ciencias de la Comunicación y Diseño
Presente

ASUNTO: Entrega de Reporte Final de Proyecto de Investigación del Dr. Francisco de Asís López F.

Estimado Dr. Mercado:

Por este conducto me permito hacerle entrega del Informe Final del Proyecto de Investigación del Dr. Francisco de Asís López Fuentes, titulado "Análisis de arquitecturas colaborativas multi-fuente usando codificación de red".

Agradeceré girar sus apreciables instrucciones para que el Informe del Dr. López Fuentes sea sometido al Consejo Divisional para su aprobación.

Se envía carta del Dr. López Fuentes así como su Informe Final en forma digital, vía correo electrónico.

Sin otro particular, le envío un cordial saludo.

Atentamente,

Casa abierta al tiempo

Dr. Carlos Joel Rivero Moreno
Jefe del Departamento de Tecnologías de la Información

UNIVERSIDAD AUTÓNOMA METROPOLITANA
Casa abierta al tiempo
UNIDAD CUAJIMALPA
RECIBIDO
17 ENE 2020
NOMBRE Y FIRMA: Perla Benítez
SECRETARÍA ACADÉMICA DE CIENCIAS DE LA COMUNICACIÓN Y DISEÑO

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Cuajim
16 ENE 2020
SONIA
Dirección
División de Ciencias de la Comunicación y Diseño

Anexo: Lo mencionado

c.c.p.: Dra. Gloria Angélica Martínez de la Peña – Secretaria del Consejo Divisional

*CJRM*pfc*

División Ciencias de la Comunicación y Diseño

Unidad Cuajimalpa
DCCD | Jefatura del Departamento de Tecnologías de la Información
Torre III, 5to. piso. Avenida Vasco de Quiroga 4871,
Colonia Santa Fe Cuajimalpa. Delegación Cuajimalpa de Morelos,
Tel. +52 (55) 5814-6557, C.P. 05348, México, D.F.
http://dccd.cua.uam.mx

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Cuajimalpa

Comunidad académica comprometida con el desarrollo humano de la sociedad.

Ciudad de México, enero 16 de 2020

Dr. Carlos Rivero Moreno
Jefe del Departamento de Tecnologías de la Información
UAM Cuajimalpa
Presente

Por este conducto me permito entregar el Informe Final del proyecto intitulado "Análisis de arquitecturas colaborativas multi-fuente usando codificación de red". Los objetivos del proyecto se cumplieron y entre los resultados destacados se encuentran:

1 servicio social concluido
1 proyecto terminal concluido
2 artículos en conferencias internacionales IEEE
1 artículo en conferencia internacional EAI
1 artículo en revista con índice LATINDEX
1 artículo en revista con índice JCR

Sin más por el momento, le agradezco la atención a la presente.

Atentamente

Casa abierta al tiempo

Dr. Francisco de Asís López Fuentes
Profesor-investigador
Departamento de Tecnologías de la Información

UNIVERSIDAD AUTÓNOMA METROPOLITANA
Casa abierta al tiempo
UNIDAD CUAJIMALPA
RECIBIDO
16 ENE 2020
NOMBRE Y FIRMA:
DEPARTAMENTO DE TECNOLOGÍAS DE LA INFORMACIÓN

INFORME DE TERMINACIÓN DE PROYECTO DE INVESTIGACIÓN

ANALISIS DE ARQUITECTURAS COLABORATIVAS MULTI-FUENTE USANDO CODIFICACION DE RED

El Proyecto **"Análisis de arquitecturas colaborativas multi-fuente usando codificación de red"** se centró en evaluar los siguientes factores:

1. Evaluar el rendimiento de un sistema basado en multi-fuente colaborativas en ambientes multi-fuente.
2. Evaluar el rendimiento de una red con múltiples fuentes usando codificación de red.
3. Modelado matemático de una arquitectura jerárquica con nodos fuente-consumidor colaborativos.
4. Evaluación de una asignación dinámica recursos para la codificación de red.
5. Estudio de técnicas de apredizaje máquina con codificación de red.

El interés por estudiar estos factores nos ha permitido entender el desempeño de la codificación de red para diferentes arquitecturas de redes con multi-fuentes así como el uso de técnicas de aprendizaje máquina para encontrar formas más óptimas de realizar la codificación de red.

Los participantes de este proyecto fueron:

- Dr. Francisco de Asís López Fuentes (responsable del proyecto)
- Dr. Rogelio Hasimoto Beltrán (CIMAT- CONACYT)

El objetivo general que corresponde a "Investigar y analizar una arquitectura multi-fuente que permitan integrar técnicas de codificación de red y optimizar la difusión de contenidos en ambientes de redes P2P" fue alcanzado ya que se lograron evaluar los desempeños de la codificación de red en arquitecturas multi-fuentes.

Las metas planteadas en el proyecto propuesto fueron alcanzadas al obtener los siguientes resultados:

1. Redes de investigación:
Se generó un fuerte trabajo colaborativo con investigadores de diferentes centros de investigación a través de la red CONACYT Red en Sistemas y Redes de Próxima Generación, así como con alumnos interesados en la codificación de red.

2. Modelo matemático
Se propuso un modelo matemático para una arquitectura basada en árboles P2P altamente colaborativa y escalable hecha de dos componentes principales: 1) Agrupación o agrupación por pares y 2) Distribución de contenido jerárquico basado en árboles. En el paso 1), los nodos pares (receptores de contenido) se agrupan en grupos de igual tamaño mediante el uso de un algoritmo heurístico de restricción de tamaño propuesto basado en k-

means. En el paso 2), los clústeres (que se convierten en los nodos del árbol) se organizan en una sola arquitectura jerárquica basada en un árbol n-ario, en la que la raíz del árbol (Clúster raíz) es la más cercana al par fuente, mientras que los clusters intermedio y los clusters hojas se colocan en el árbol de acuerdo con su retraso de proximidad a los clusters previamente insertados.

3. Estancia de investigación corta:

Se realizaron dos estancias de investigación corta en el CIMAT-Guanajuato durante el mes de agosto de 2017 y agosto de 2018. Los gastos de hospedaje y transporte fueron cubiertos por el CIMAT o por la red temática del CONACYT Red en Sistemas y Redes de Próxima Generación.

También el alumno Javier Mendoza Almanza hizo una estancia de una semana en el CIMAT en la ciudad de Guanajuato durante 2017. Los gastos de viáticos fueron cubiertos por el CIMAT, mientras que los gastos de transporte los cubrió la UAM.

4. Formación de recursos humanos:

El proyecto permitió involucrar a alumnos de licenciatura en Tecnologías y Sistemas de Información. Se realizaron proyecto terminal de licenciatura donde participaron los alumnos:

Javier Mendoza Almanza
Raúl Ortega Vallejo – inició su proyecto terminal

Un servicio social donde participó el alumno:

Javier Mendoza Almanza

5. Artículos de investigación:

Los resultados de este proyecto se publicaron en 2 conferencias internacionales de IEEE, una conferencia internacional EAI, y dos revistas indizadas. Una revista indizada en JCR con factor de impacto de 2.39 (2018) y otra indizada en Latindex:

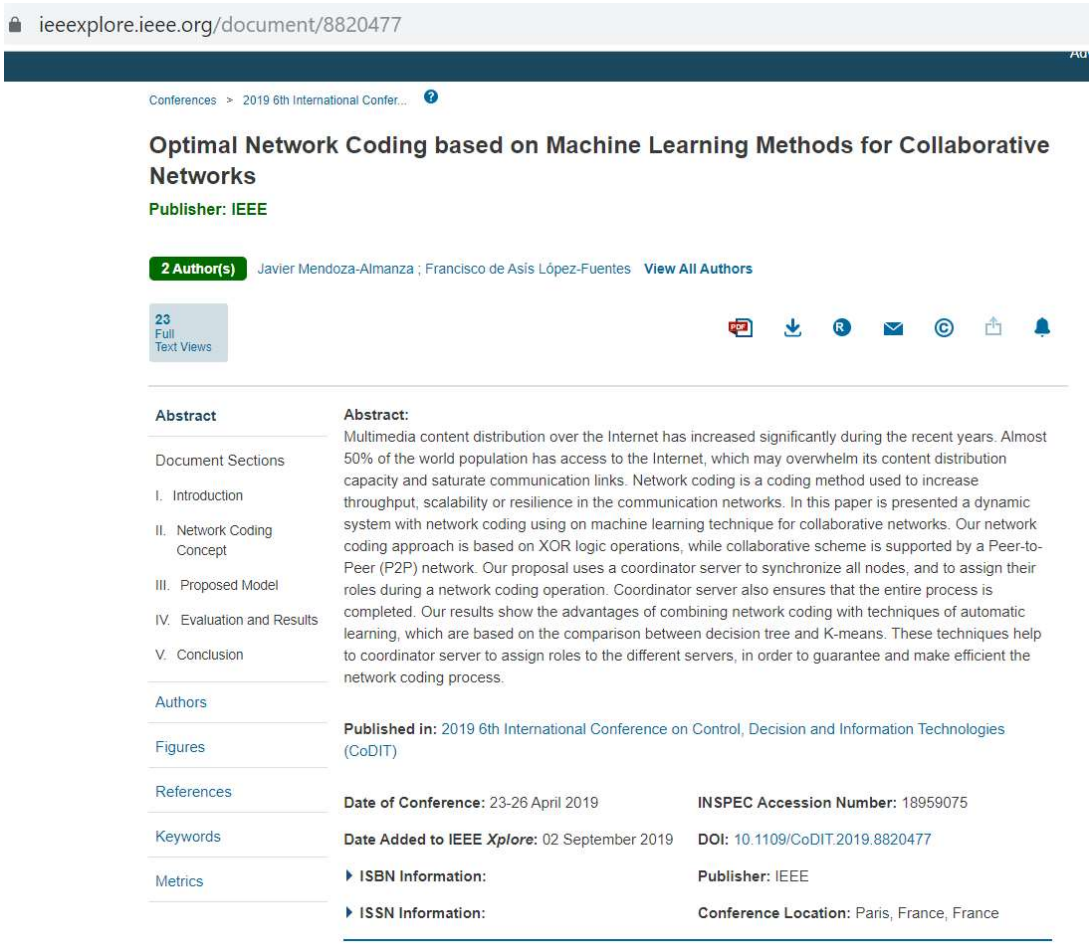
1. **López-Fuentes, F. A.** and Mendoza-Almanza, J.; "Optimal Network Coding based on Machine Learning Methods for Collaborative Networks," 6th *IEEE Int. Conf. on Control, Decision and Information Technology,* Paris, France, April 2019.
2. Hasimoto-Beltran, R**., López-Fuentes, F.A.** and Vera-Lopez, M.; "Hierarchical P2P architecture for efficient content distribution," *Peer-to-Peer networking and applications, Springer*, August 2018, Online ISSN 1936-6450. **(JCR, FI = 2.39)**
3. **López-Fuentes, F. A.** and Mendoza-Almanza, J.; "Dynamic Network Coding for Collaborative Multi-source System," *9th IEEE Annual Information Technology,*

Electronics, and Mobile Communication Conference, Vancouver, BC, Canada, November 2018.

4. Mendoza-Almanza J., **López-Fuentes F. A.** and Hasimoto R.; **"**Practical Network Coding for Multi-source Scenarios," *Book: Smart Technology, Chapter 14, LNICST, Springer Nature*, Chapter DOI: 10.1007/978-3-319-73323-4_14, 2018.
5. Mendoza-Almanza J. and **López-Fuentes, F. A**.; "Collaborative Multi-source Scheme for Multimedia Content Distribution," *Research in Computing Science*, Vol. 127, pp. 51-57, 2016, ISSN 1870-4069. (**latindex**)

Se anexan los resumenes de los artículos como fueron publicados por las editoriales IEEE y Springer:

López-Fuentes, F. A. and Mendoza-Almanza, J.; "Optimal Network Coding based on Machine Learning Methods for Collaborative Networks," 6th *IEEE Int. Conf. on Control, Decision and Information Technology,* Paris, France, April 2019.


ieeexplore.ieee.org/document/8820477

Conferences > 2019 6th International Confer...

Optimal Network Coding based on Machine Learning Methods for Collaborative Networks

Publisher: IEEE

2 Author(s) Javier Mendoza-Almanza ; Francisco de Asís López-Fuentes **View All Authors**

23 Full Text Views

Abstract

Document Sections

I. Introduction

II. Network Coding Concept

III. Proposed Model

IV. Evaluation and Results

V. Conclusion

Authors

Figures

References

Keywords

Metrics

Abstract:
Multimedia content distribution over the Internet has increased significantly during the recent years. Almost 50% of the world population has access to the Internet, which may overwhelm its content distribution capacity and saturate communication links. Network coding is a coding method used to increase throughput, scalability or resilience in the communication networks. In this paper is presented a dynamic system with network coding using on machine learning technique for collaborative networks. Our network coding approach is based on XOR logic operations, while collaborative scheme is supported by a Peer-to-Peer (P2P) network. Our proposal uses a coordinator server to synchronize all nodes, and to assign their roles during a network coding operation. Coordinator server also ensures that the entire process is completed. Our results show the advantages of combining network coding with techniques of automatic learning, which are based on the comparison between decision tree and K-means. These techniques help to coordinator server to assign roles to the different servers, in order to guarantee and make efficient the network coding process.

Published in: 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)

Date of Conference: 23-26 April 2019
INSPEC Accession Number: 18959075
Date Added to IEEE *Xplore*: 02 September 2019
DOI: 10.1109/CoDIT.2019.8820477
ISBN Information:
Publisher: IEEE
ISSN Information:
Conference Location: Paris, France, France

La asistencia a esta conferencia fue soportada completamente por el Departamento de Tecnologías de la Información- UAM Cuajimalpa.

Hasimoto-Beltran, R**., López-Fuentes, F.A.** and Vera-Lopez, M.; "Hierarchical P2P architecture for efficient content distribution," *Peer-to-Peer networking and applications, Springer*, August 2018, Online ISSN 1936-6450.

Springer Link

Peer-to-Peer Networking and Applications
July 2019, Volume 12, Issue 4, pp 724–739 | Cite as

Hierarchical P2P architecture for efficient content distribution

Authors | Authors and affiliations

Rogelio Hasimoto-Beltran, Francisco de Asís Lopez-Fuentes, Misael Vera-Lopez

Article
First Online: 01 August 2018

184 Downloads
1 Citations

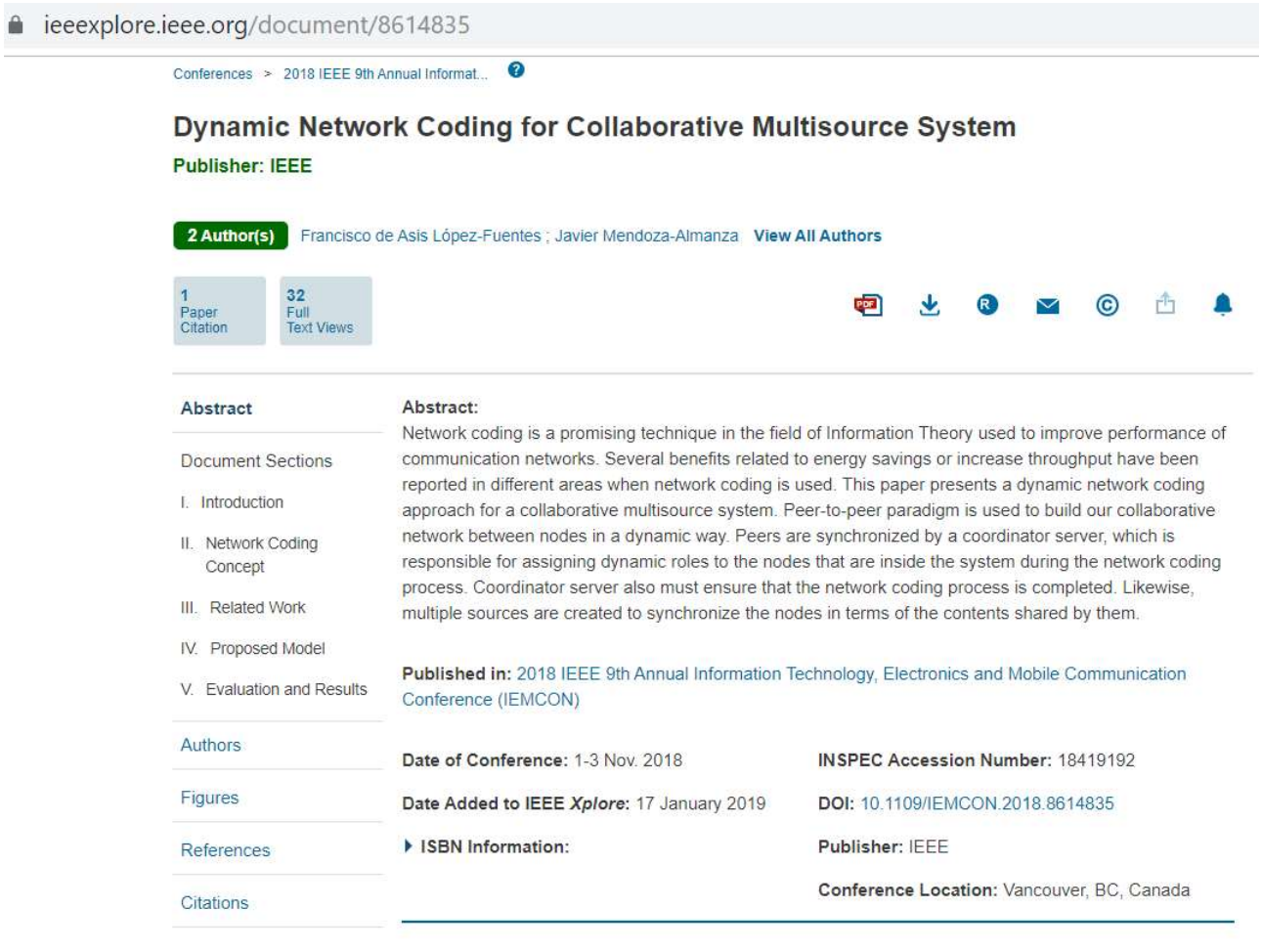
Abstract

Mutualcast is a one-to-many (peer-to-peer) scheme for content distribution that maximizes the overall throughput during a broadacast session. It is based on a fully-connected graph (full mesh topology), which introduces benefits such as robustness or simultaneous transmission from/to multiple devices. The main disadvantage of Mutualcast is scalability; it is constraint to a small P2P group for content distribution. In this paper, we make Mutualcast scalable. We propose a highly collaborative and scalable P2P tree-based architecture made of two main components: 1) Peer grouping or clustering and 2) Hierarchical tree-based content distribution. In step 1), peer nodes (content receivers) are grouped into equal-size clusters by using a proposed heuristic size-constrained algorithm based on k-means. In step 2), clusters (which become the nodes of the tree) are organized into a single hierarchical n-ary tree-based architecture, in which the root of the tree (Root Cluster) is the one closest to source peer, while intermediate and leaf clusters are positioned in the tree according to their delay-proximity to previously inserted clusters. During content distribution, the root cluster receives the blocks of content before any other cluster in the tree and directly from (and only from) the source peer; blocks are then passed on to the next hierarchical level down the tree in order (higher levels of the tree receive the content before lower levels). Inter-clusters and intra-clusters content distribution is performed concurrently and takes into account peers upload/download capacities to relay blocks of content. The evaluation of our hierarchical P2P architecture concentrates on the following metrics: scalability of the systems, overall end-to-end delay distribution, and efficient cluster size. Finally, our architecture is compared against two well-known P2P technologies in the literature, Super-Peer and Kademlia.

Keywords

Peer-to-peer networks | Video multicast | Content distribution | Clustering

López-Fuentes, F. A. and Mendoza-Almanza, J.; "Dynamic Network Coding for Collaborative Multi-source System," *9th IEEE Annual Information Technology, Electronics, and Mobile Communication Conference,* Vancouver, BC, Canada, November 2018.



ieeexplore.ieee.org/document/8614835

Conferences > 2018 IEEE 9th Annual Informat...

Dynamic Network Coding for Collaborative Multisource System

Publisher: IEEE

2 Author(s) Francisco de Asis López-Fuentes ; Javier Mendoza-Almanza **View All Authors**

1 Paper Citation

32 Full Text Views

Abstract

Document Sections

I. Introduction

II. Network Coding Concept

III. Related Work

IV. Proposed Model

V. Evaluation and Results

Authors

Figures

References

Citations

Abstract:
Network coding is a promising technique in the field of Information Theory used to improve performance of communication networks. Several benefits related to energy savings or increase throughput have been reported in different areas when network coding is used. This paper presents a dynamic network coding approach for a collaborative multisource system. Peer-to-peer paradigm is used to build our collaborative network between nodes in a dynamic way. Peers are synchronized by a coordinator server, which is responsible for assigning dynamic roles to the nodes that are inside the system during the network coding process. Coordinator server also must ensure that the network coding process is completed. Likewise, multiple sources are created to synchronize the nodes in terms of the contents shared by them.

Published in: 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)

Date of Conference: 1-3 Nov. 2018

Date Added to IEEE *Xplore*: 17 January 2019

ISBN Information:

INSPEC Accession Number: 18419192

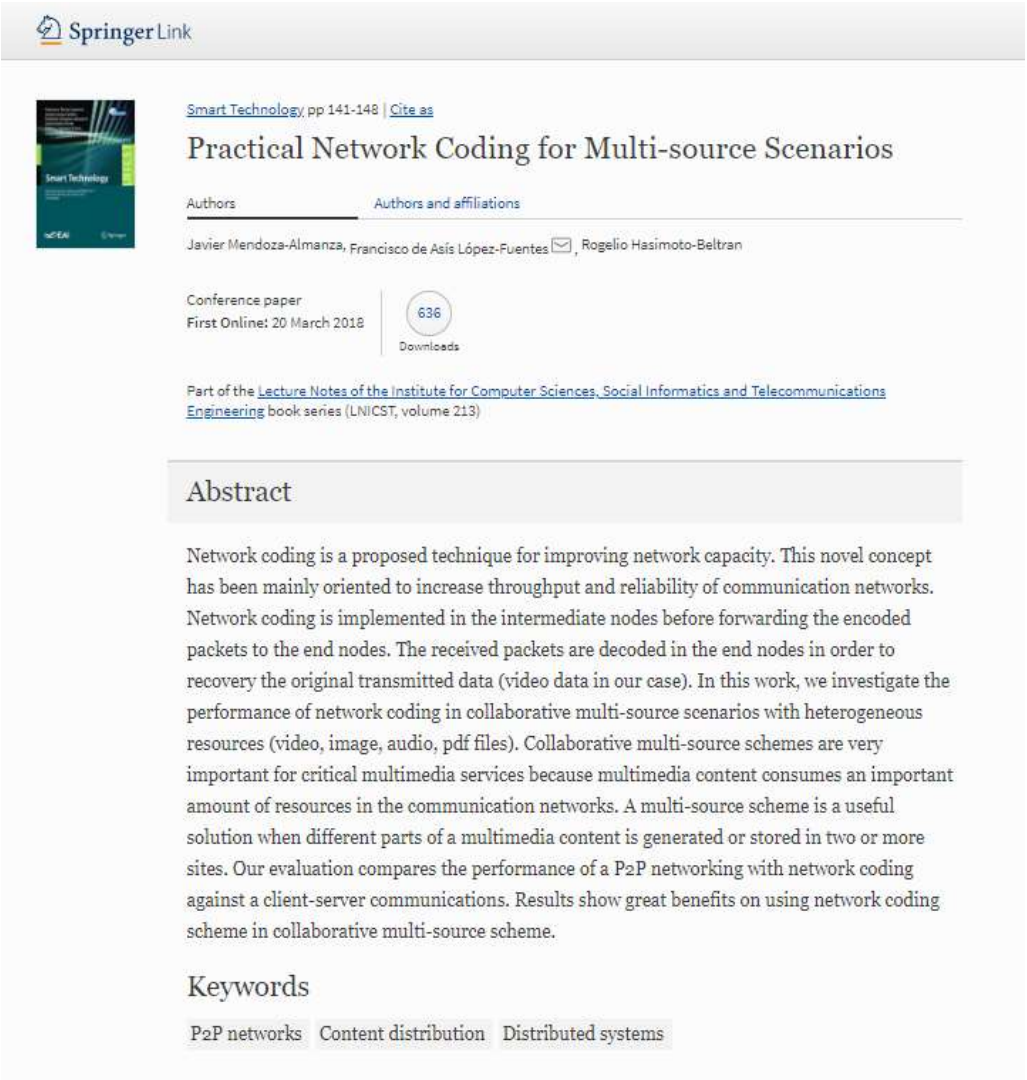
DOI: 10.1109/IEMCON.2018.8614835

Publisher: IEEE

Conference Location: Vancouver, BC, Canada

La asistencia a esta conferencia fue soportada completamente por el Departamento de Tecnologías de la Información- UAM Cuajimalpa.

Mendoza-Almanza J., **López-Fuentes F. A.** and Hasimoto R.; **“**Practical Network Coding for Multi-source Scenarios,” *Book: Smart Technology, Chapter 14, LNICST, Springer Nature*, Chapter DOI: 10.1007/978-3-319-73323-4_14, 2018.



SpringerLink

Smart Technology, pp 141-148 | Cite as

Practical Network Coding for Multi-source Scenarios

Authors | Authors and affiliations

Javier Mendoza-Almanza, Francisco de Asís López-Fuentes ✉, Rogelio Hasimoto-Beltran

Conference paper
First Online: 20 March 2018

636 Downloads

Part of the Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering book series (LNICST, volume 213)

Abstract

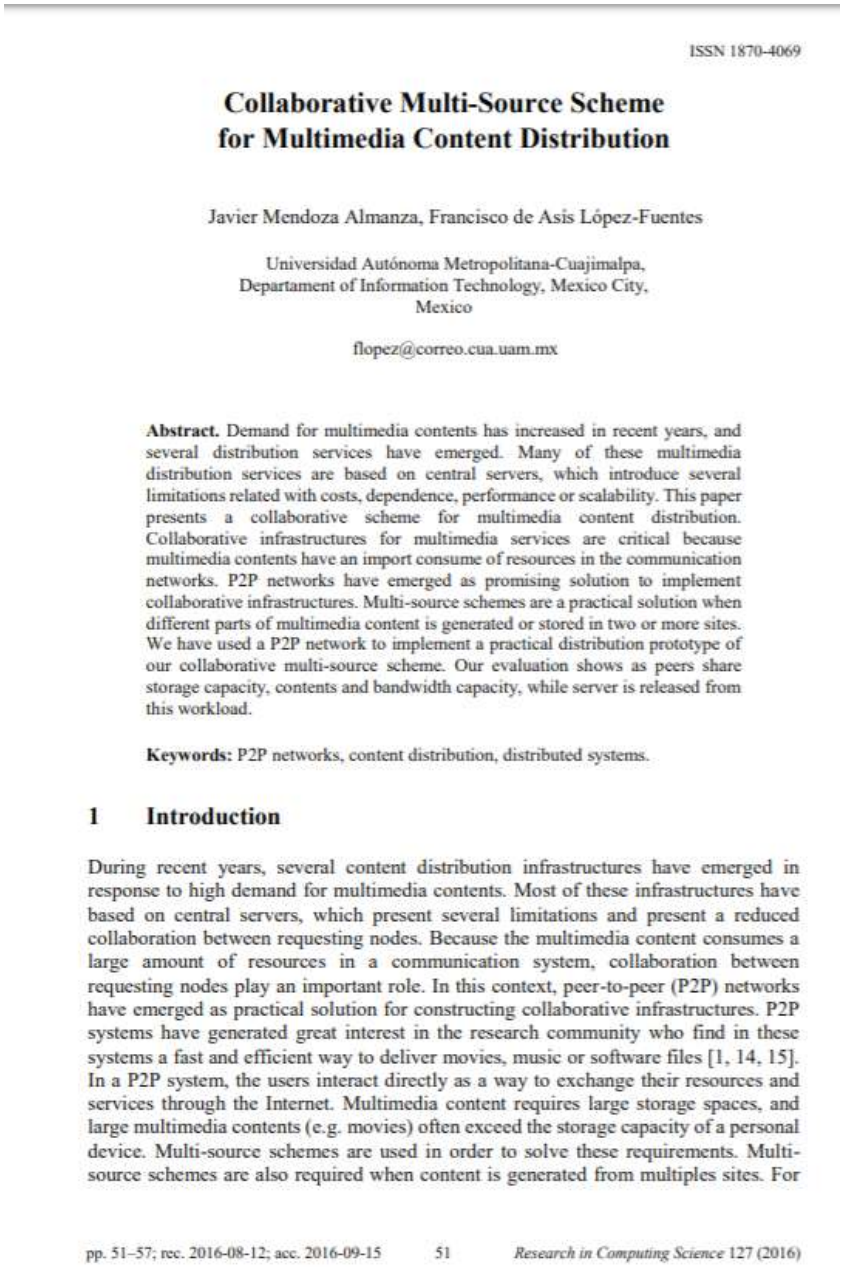
Network coding is a proposed technique for improving network capacity. This novel concept has been mainly oriented to increase throughput and reliability of communication networks. Network coding is implemented in the intermediate nodes before forwarding the encoded packets to the end nodes. The received packets are decoded in the end nodes in order to recovery the original transmitted data (video data in our case). In this work, we investigate the performance of network coding in collaborative multi-source scenarios with heterogeneous resources (video, image, audio, pdf files). Collaborative multi-source schemes are very important for critical multimedia services because multimedia content consumes an important amount of resources in the communication networks. A multi-source scheme is a useful solution when different parts of a multimedia content is generated or stored in two or more sites. Our evaluation compares the performance of a P2P networking with network coding against a client-server communications. Results show great benefits on using network coding scheme in collaborative multi-source scheme.

Keywords

P2P networks | Content distribution | Distributed systems

La asistencia a esta conferencia en Monterrey NL. fue soportada en parte por la Red Temática CONACYT Red en Sistemas y Redes de Próxima Generación y por la Coordinación de la Licenciatura en Tecnologías y Sistemas de Información.

Mendoza-Almanza J. and **López-Fuentes, F. A**.; "Collaborative Multi-source Scheme for Multimedia Content Distribution," *Research in Computing Science*, Vol. 127, pp. 51-57, 2016, ISSN 1870-4069.



ISSN 1870-4069

Collaborative Multi-Source Scheme for Multimedia Content Distribution

Javier Mendoza Almanza, Francisco de Asis López-Fuentes

Universidad Autónoma Metropolitana-Cuajimalpa,
Departament of Information Technology, Mexico City,
Mexico

flopez@correo.cua.uam.mx

Abstract. Demand for multimedia contents has increased in recent years, and several distribution services have emerged. Many of these multimedia distribution services are based on central servers, which introduce several limitations related with costs, dependence, performance or scalability. This paper presents a collaborative scheme for multimedia content distribution. Collaborative infrastructures for multimedia services are critical because multimedia contents have an import consume of resources in the communication networks. P2P networks have emerged as promising solution to implement collaborative infrastructures. Multi-source schemes are a practical solution when different parts of multimedia content is generated or stored in two or more sites. We have used a P2P network to implement a practical distribution prototype of our collaborative multi-source scheme. Our evaluation shows as peers share storage capacity, contents and bandwidth capacity, while server is released from this workload.

Keywords: P2P networks, content distribution, distributed systems.

1 Introduction

During recent years, several content distribution infrastructures have emerged in response to high demand for multimedia contents. Most of these infrastructures have based on central servers, which present several limitations and present a reduced collaboration between requesting nodes. Because the multimedia content consumes a large amount of resources in a communication system, collaboration between requesting nodes play an important role. In this context, peer-to-peer (P2P) networks have emerged as practical solution for constructing collaborative infrastructures. P2P systems have generated great interest in the research community who find in these systems a fast and efficient way to deliver movies, music or software files [1, 14, 15]. In a P2P system, the users interact directly as a way to exchange their resources and services through the Internet. Multimedia content requires large storage spaces, and large multimedia contents (e.g. movies) often exceed the storage capacity of a personal device. Multi-source schemes are used in order to solve these requirements. Multi-source schemes are also required when content is generated from multiples sites. For

La asistencia del alumno Javier Mendoza Almanza a esta conferencia en Silao Gto. fue soportada por la Coordinación de Tecnología y Sistemas de Información.

2019 6th International Conference on Control, Decision and
Information Technologies (CoDIT'19) | Paris, France / April 23-26, 2019

Optimal Network Coding based on Machine Learning Methods for Collaborative Networks

Javier Mendoza-Almanza, Francisco de Asís López-Fuentes

Abstract— **Multimedia content distribution over the Internet has increased significantly during the recent years. Almost 50% of the world population has access to the Internet, which may overwhelm its content distribution capacity and saturate communication links. Network coding is a coding method used to increase throughput, scalability or resilience in the communication networks. In this paper is presented a dynamic system with network coding using on machine learning technique for collaborative networks. Our network coding approach is based on XOR logic operations, while collaborative scheme is supported by a Peer-to-Peer (P2P) network. Our proposal uses a coordinator server to synchronize all nodes, and to assign their roles during a network coding operation. Coordinator server also ensures that the entire process is completed. Our results show the advantages of combining network coding with techniques of automatic learning, which are based on the comparison between decision tree and K-means. These techniques help to coordinator server to assign roles to the different servers, in order to guarantee and make efficient the network coding process.**

I. Introduction

Multimedia content distribution has gained popularity during the last years. This type of content is generated from different means such as social networks, mobile devices, etc., and different content distribution infrastructures have emerged to deal with this high demand. Multimedia consumes a large amount of resources in the system infrastructure and collaboration between nodes is very important. However, most of current infrastructures are centralized and collaboration between nodes is limited. On the other hand, peer-to-peer (P2P) networks have gained very popularity during last decades. This fact is due to these networks present high performance characteristics such as dynamicity, scalability, multiplicity, efficient content distribution and ability to search effectively [10].

In this context, a system can fragment and locate a large multimedia content in different nodes or peers, and users can retrieve the multimedia content from multiple sources. Multi-source concept also helps to alleviate the unpredictability congestion in the Internet, and it is an alternative to provide smooth video delivery [1], [2]. This paper combines machine learning techniques with network coding in a collaborative network with multiple sources to improve video transmission.

F. A. López-Fuentes is with the Information Technology Department, Universidad Autónoma Metropolitana-Cuajimalpa, Mexico City, 05300, Mexico (corresponding author to provide phone: 55-58145500; fax: 303-555-5555; e-mail: flopez@correo.cua.uam.mx).

J. Mendoza-Almanza was with Universidad Autónoma Metropolitana-Cuajimalpa, Mexico City, 05300, Mexico (e-mail: javier-lvr09@hotmail.com).

To reach an acceptable video quality, a system should offer high data rate, low-latency and high throughput, and network coding can help in these tasks. Network coding is done in the intermediate nodes [7], [8] and [9]. In this work, P2P networks are used as collaborative platform to reach distribution of load and duties between all participating nodes. We aim to implement a balanced network coding between all participating peer. To reach this goal, we extend cooperation between participating peers by using their processing and uploading capabilities instead of limiting the cooperation of peers to only their storage capacity. On the other hand using network coding, the intermediate nodes send out packets that are combinations of previously received packets instead of simply forwarding them. These packet manipulations are linear operations over elements of a finite field. Traditionally, intermediate nodes perform network coding in a static way. This means that only specific nodes can do network coding. In contrast, in a dynamic approach any node in the system can do network coding. However, simple dynamic approach has different disadvantages. For example, some of the current systems have a coordination server that allows for balancing and different roles in the nodes of the network, but this is not useful when the system has a large number of processes at the same time and there is not enough active nodes. When in a system the coordinator server does not have enough nodes, it should wait for nodes finish their current role to give they a new one. The system does not take into account the capacity of the nodes, and these do not take full advantage of the resources since in many cases the nodes can play different roles at the same time, which would imply an improvement over time of the system. In this paper we present an extended version of our work presented in [17]. In this case we explore machine learning techniques as a way to reach an optimal allocation of network coding operations in collaborative networks. We believe that a machine learning model can predict the performance of the nodes in order to improve the performance of the system.

To apply machine learning techniques in a scheme with network coding is required that all participating nodes take different roles during a network coding operation. These roles are: receiver, encoder, sender, relay node. To reach a dynamic participation of all nodes our proposal considers a collaborative approach supported by a P2P network. In this work, two models of machine learning have been used which are decision tree and k-means. The main objective is to predict the number of roles that a node can execute at the same time. As a second objective, we would like to know the ideal capabilities required by a node to play a certain number of roles at the same time in terms of its RAM, CPU and disk in order to take advantage of the resources of the nodes and

978-1-7281-0521-5/19/$31.00 ©2019 IEEE

avoid saturating them, which would make the process of network coding. We evaluate our proposed scheme using a prototype implemented in Linux where the communications between all nodes are established via TCP (Transmission Control Protocol). Our proposal is done for network coding based on XOR operation.

The rest of this paper is organized as follows. Section II presents the main idea of network coding. Then, we describe our proposed network coding model in Section III. An implementation of our proposed model using machine learning techniques over a P2P network is presented in Section IV. In this section we also compare performance of our proposal using decision tree and k-means techniques. Conclusions are given in Section V.

II. Network Coding Concept

Data communication can be more efficient if data is sent in form of scrambled packets instead of sending plain data. Scrambled operation is done in a completely random fashion. Network coding was introduced by Ahlswede at al [3] as a technique for the diffusion of the information in the field of information theory. This technique introduces several benefits in the communication networks such as throughput enhancement, increase of capacity, robustness, tomography and security [6-7, 11]. Network coding employs coding at the intermediate nodes to increase the flow of packets without exceeding the link capacity. To explain network coding concept let's to consider the butterfly network shown in Fig. 1. In this scheme there are a source node and two receiver nodes. Figure 1a shows the capacity of each edge is 1. We can observe that the value of the maximum flow of S to any receiver, either $R1$ or $R2$ is equal to two. In Fig. 1b source S sends two bits, $b1$ to R1 and $b2$ to $R2$ simultaneously. In this scheme, intermediate node 3 only replicates and sends out the bits received from nodes 1 and 2. We can see that link between node 3 and node 4 needs two time units to send b1 and b2 to node 4. On the other hand, Fig. 1c shows the communication network with network coding, where operator $\oplus$ is used to denote sum module 2. Thus, the receiver $R1$ can recover the two bits, $b1$ and $b2$, except that $b2$ must be retrieved from $b1 \oplus b2$. Similarly, $R2$ can recover the two bits. In this case, network coding is applied at node 3. Another important point is that the multicast rate increases, because in traditional transmission the rate is 1 bit/time unit, whereas applying network coding the rate increases to 2 bits/time unit.

The most common benefits of using network coding in a communication network are [6-7, 10]: Throughput enhancement can be achieved by sending more information using fewer packets. Reduction of packet loss is possible because overflow, link outage and collision are avoided by applying network coding at intermediate nodes. Increase of security can be reached by sending scrambled data. Thus, attackers are not able to find out the original data. Also. Network coding helps to improve robustness in the communication networks.

Network coding concept can be applied at different layers of TCP/IP model. In this paper, network coding is applied in the application layer, which provides services for application programs. Multicast protocol in this layer is called

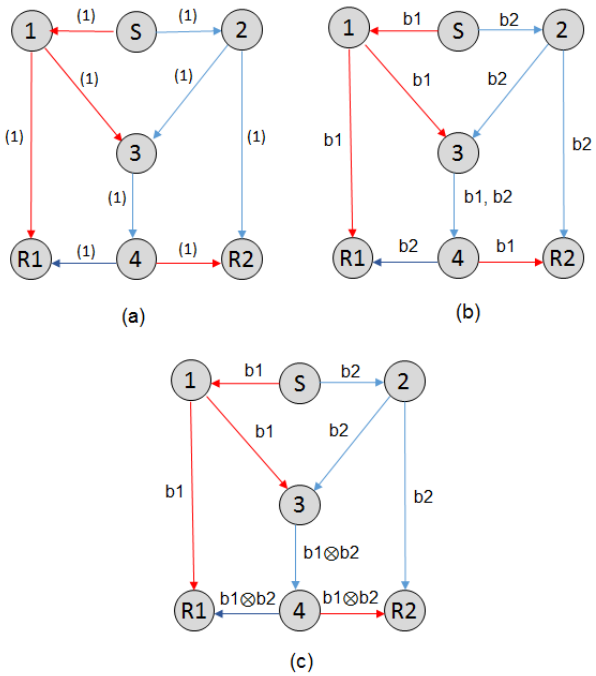


Figure 1. Case for a communication network. a) Capacity of the links, b) Traditional approach and c) approach using network coding.

Application Layer Multicast (ALM) protocol, and it is implemented at end hosts instead of network routers. However, ALM protocol is affected by delay, throughput and security issues [14]. To deal these issues, network coding is implemented at end hosts as an application program. Thus, our network coding solution does not require any infrastructure support, and can easily be deployed with ALM protocol over the Internet.

Several applications have integrated network coding to improve their performance. Some example are sensor networks, wireless relay networks, wireless local network, MANETs, ad-hoc networks and wireless mesh networks [12]. In most of these cases, network coding is used to provide reliable broadcasting, efficient data dissemination, file sharing, multimedia streaming and recovery data.

III. Proposed Model

In this section we introduce our proposed model. As background of our model, we give a briefly overview about different schemes involved with our proposal. Our collaborative multi-source system uses different servers and peers to distribute different types of files.

A. Multi-source

In our collaborative multi-source model each source distributes its content to the requesting peers. This scenario is shown in figure 2. Our solution assumes that sources are independent of each other, and each source distribute a specific type of files. For example, source S1 distributes video files, while S2 distributes music files, S3 distributes

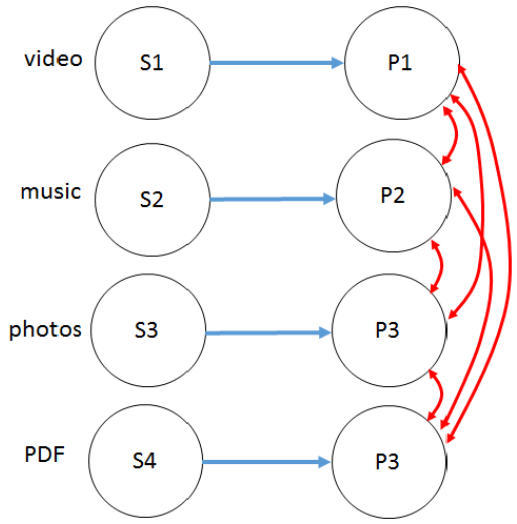


Figure 2. A collaborative multi-source scheme

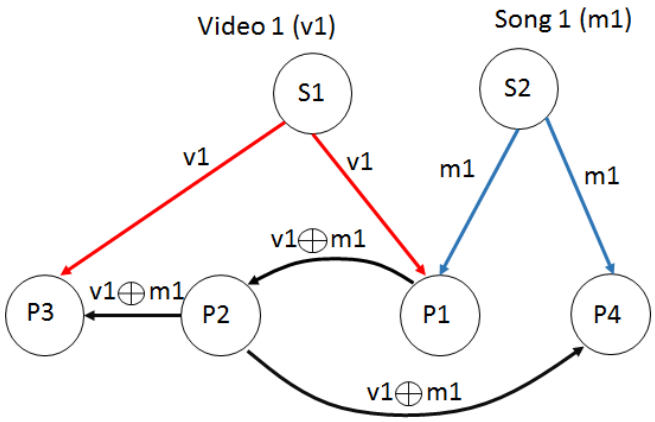


Figure 3. Network coding in a multi-source scenario. Peer P1 does network coding, and P3 and P4 are the sink nodes.

photos files and S4 distributes PDF files. We use this scheme to implement network coding. In Figure 2, we can see that each peer establishes communication with only a source, and all peers (P1, P2, P3 and P4) establish communication one another, and all peers can share the files received from the sources. In this way, a peer can act as a server when it distributes the received file from the original source, and as a client when it requests a file from another peer. In server mode, a peer must respond to all received requests. In each peer is stored the IP address of all nodes and their shared files in a matrix. When a peer (in server mode) receives a request, it creates a thread to attend the request. For each request a thread is created to respond to all at the same time. This thread establishes a communication with the peer that wants to share files or synchronize with the server. Each node uses different matrices to store information of the connected nodes and their shared files. We use the IP address of the peers and the name of shared files to organize information in these matrices. These matrices are synchronized with the different sources when the peer wants some content from another peer. A peer creates different threads to request a file. These threads are used to synchronize the matrices and to update its information (IP address and shared files) only of the active nodes in the system. After the matrices have been updated, the requesting peer establishes via threads a connection with peers where the requested files are available. Different threads can be created by a requesting peer to receive different files from different supplier peers. Also, a peer can distribute a file to all peers that request it. A more detailed explanation about this multi-source architecture can be found in [4].

B. *Static network coding*

A collaborative multi-source architecture based on P2P networks can perform network coding. In this section, we briefly explain a static network coding previously worked and presented in [5]. In this case, source and peers are organized as is shown in figure 3. We use a traditional network coding approach based on the butterfly concept. We have called this solution as static network coding, because each peer in the architecture has only a specific task to do during all the network coding process. In figure 3, peer P1 receives a video v1 from source S1 and a song m1 from source S2. After this, peer P1 reads bit by bit from both files and applies the XOR operation to create an encoded file. Sink peers P3 and P4 receive the encoded file and create a thread to one of the source peers to request an original file. Thus, P3 requests video v1 to source S1 to retrieve song m1 from v1⊕m1, and P4 requests song m1 to source S2 to retrieve video v1 from v1⊕m1, too. For each bit that reads from both files, the peers apply the XOR operation and the result will be stored in a new file. Different experiments presented in [5] show that a system using traditional network coding can improve its bandwidth gain around 33% compared to a system where network coding is not used.

C. *Dynamic network coding*

Traditional network coding presents some limitations because encoder role cannot be shared between all peers in a collaborative way. This fact introduces saturation of tasks in a unique peer. To avoid this scenario in [17] is presented a dynamic network coding approach. This dynamic scheme incorporates a coordinator server, which assigns different roles to the peers in order to avoid the saturation of encoding tasks in a unique peer. Coordinator server has two matrices, first matrix stores the address of all peers within the network, while second matrix stores the roles of all peers during the network coding process. Each peer has six different roles which are source1, source2, encoder, distributor, receiver1 and receiver2, which makes this matrix, is to save by process which nodes are occupying that role, to avoid saturating them. Figure 4 shows these types of roles for each peer. The operation of dynamic network coding is as follows. After to receive a request from a peer, coordinator server automatically assigns roles to all active peers in the system. To run network coding is important that coordinator server define which peers will be the receiver (sink) peers. Once the sources and receivers are defined, the coordinador server obtains the nodes that are not present in the process and makes two selections of random way to obtain two IP addresses. Then, a peer is assigned as encoder, while other

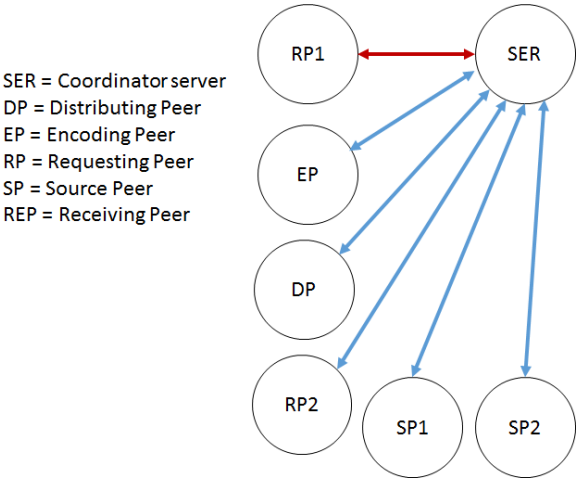


Figure 4. Coordinator server interacts with nodes to define its roles during a dynamic network coding operation.

peer assigns is assigned as a distributor. After all necessary peers have been obtained, the coordinator server stores their IP addresses in a process matrix and creates threads to sources to notify their role. When all peers have finished their assigned role, coordinator server destroys all communication threads all peers are released.

D. Machine Learning

Machine learning has been widely used to solve several challenges in many fields. In this section, the models to be used are briefly explained. For the sake of completeness, we provide a brief introduction of supervised machine learning for classification here.

D.1. Decision tree

A decision tree is an analytical method that proposes specific alternatives that allow the best decision making, taking into account benefits, risks, costs and all those variables that are wished to be taken into account during the analysis. In machine learning, a decision tree is a learning model that is based on a supervised learning algorithm which allows classifying or performing the regression task.

Decision trees have a root node that can be fragmented into two or more nodes and then those nodes into two or more to reach the leaves. The implementation of a tree with few variables is simple. However, what happens if the variables are 15 or more, we would have to make hundreds or thousands of combinations and it is very complex. Due to the complexity of finding a solution, machine learning is implemented because it allows us to obtain an optimal decision tree for the most accurate decision making from a probabilistic point of view.

At the beginning, a decision tree must take into account all its variables and define which of them are the most important to define a scheme and obtain optimal results. For this, each subdivision between the different possible trees must be evaluated and, from these, the root node and, subsequently, the subsequent ones. The algorithm measures the predictions obtained in different ways and evaluates them to obtain the best scheme. Different metrics are used to provide a measure of the quality of the division, among the best known we can mention the gain of information and the Gini impurity [15]. The gain of information seeks that the categorical variables estimate the information provided by each one of this, in such a way that the uncertainty value of which the entropy is defined can be obtained. On the other hand, impurity Gini is used for variables with continuous values since what is sought is to measure the degree of impurity, that is, how disordered or mixed the nodes are.

D.2. K-means

On the other hand, K-means is an unsupervised clustering algorithm, which seeks to find the K clusters (groups) among the different input data [16]. K-means algorithm works iteratively to assign a point, which is a cluster based on the characteristics of the database. K-means algorithm allows finding centroids of each of the clusters which are used to classify new samples. These also allow finding the labels for the training data set, that is, each label be-longs to each one of the formed clusters.

In our model, the clusters are defined in an organic way, and adjust their position in each iteration until the algorithm converges. After the centroids are found, we analyze the unique characteristics among them, that is to say, what different characteristics present a cluster respect to others. As in the decision tree, k-means algorithm looks to predict how many processes can be executed in each node based on the same database and with the RAM, DISK and CPU variables. The K-means algorithm, unlike the decision tree, is an algorithm whose inputs must be pure numerical values. This algorithm operates in our architecture with network coding as following. Initially, it is necessary to specify how many nodes are to be created. Once this number has been defined the algorithm assigns random coordinates to each centroid.

Then, an iterative process is performed in each centroid, where each iteration has two steps. First step assignments the data while the second step updates the centroid. The assignment of data consists in that for each tuple of the database it is assigned to the nearest centroid based on the square Euclidean distance. The update of the centroid in this step means that current centroids of each cluster are recalculated based on the average of all the points assigned in the first step. These steps are done until any of the following rules are not complied:

- there are no changes in the points assigned to each group,
- the maximum number of iterations is reached,
- the sum of their Euclidean distances is reduced.

IV. Evaluation and Results

To evaluate our system with machine learning techniques, we have implemented the decision tree and the k-means models in the Python language. Our goal is to

predict the number of functions that a node can execute at the same time and the ideal capabilities for each node to perform several roles at the same time. Initially, system creates a database to store information about each node. We analyze the performance of each node in operation with respect to its processing capacity, RAM and disk. The database stores 9 variables which are the IP address, Role (which corresponds 1 = source, 2 = receiver, 3 = encoder and 4 = distributor), RAM (percentage of RAM used during the process), CPU (percentage of CPU used during the process and Disk (percentage of disco used during the process), the number of processes performed by the node, processor frequency, RAM capacity and hard disk capacity available in each node. Using information from RAM, CPU and disk, the system tries to predict the number of roles that can be implemented in each node. We can display information of the database for each attribute of these nodes. Figure 5 shows how each node is classified with respect to the number of processes; we can see eight colors which represent the different numbers of processes. The axes represent the variables of disk, CPU and RAM, while the image shows how the nodes are scattered. Based on the number of processes the nodes can execute and taking the variables already mentioned. We evaluate the machine learning methods for dynamic network coding. Our first experiment evaluates dynamic network coding using the decision tree method. In this case, the precision of algorithm is 1. The confusion matrix for the decision tree is shown in figure 6. From the confusion matrix we can observe that the prediction was correct for all the nodes. The average absolute error for dynamic network coding using a decision tree is 0.0. Results obtained from this experiments show that the first variable considered to classify is the RAM. Our second experiment applies the K-means method for dynamic network coding. In this case, we need to determine the K value. Results from this experiment obtain a K value of 6. Thus, K-means algorithm is executed with 6 clusters in order to obtain the labels and centroids. We have plotted our results obtained from K-means algorithm in 3D graphic using different colors for each clusters in order to observe some difference between them. A star indicates the centroids of each cluster. Our obtained 3D graphic is shown in figure 7. We can observe how the K-Means algorithm with 6 clusters has grouped the different nodes by the number of processes. Also, we can know the number of users in each cluster. We determinate that K-means algorithm has a precision of 0.8855932203.

Similar to decision tree case, we obtain the confusion matrix of figure 8 using K-means algorithm. We can observe that the model classified well, in the confusion matrix for nodes with 1 process, for nodes with 2 processes, correctly classify 51 of 61, for nodes with 3 processes correctly classify 52 of 80, for nodes with 4 processes classify all correctly, for nodes with 5 process classify all correctly, for nodes with 6 processes correctly classify 11 of 16, for nodes with 7 process correctly classify 9 of 13 and for nodes with 8 processes 1 correctly classify 4 of 11.

On the other hand, information obtained from the correlation matrix shows that the classification was mostly

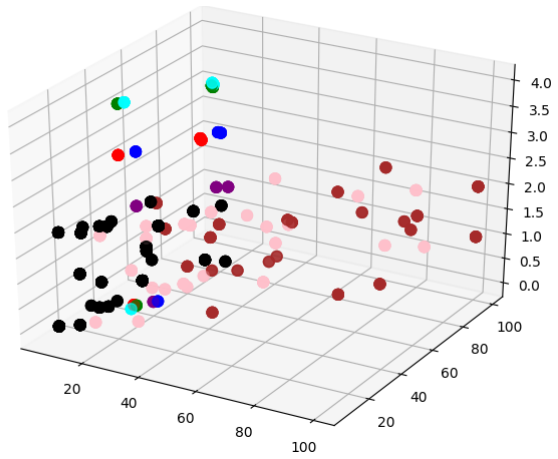


Figure 5. Interaction between processes for dynamic network coding

```
[272   0   0   0   0   0   0   0]
[  0  61   0   0   0   0   0   0]
[  0   0  80   0   0   0   0   0]
[  0   0   0  10   0   0   0   0]
[  0   0   0   0   9   0   0   0]
[  0   0   0   0   0  16   0   0]
[  0   0   0   0   0   0  13   0]
[  0   0   0   0   0   0   0  11]]
```

Figure 6. Confusion matrix for dynamic network coding using decision tree

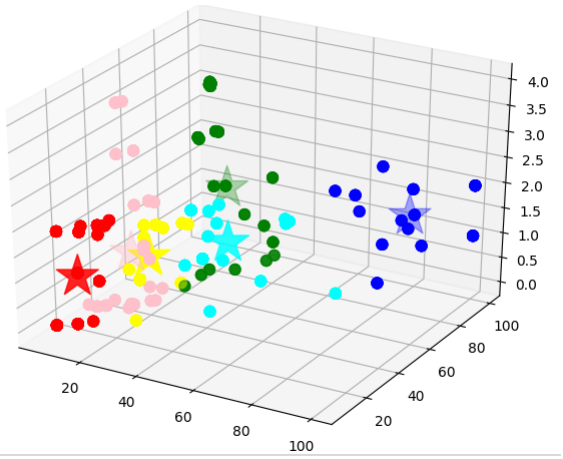


Figure 7. Centroids obtained after running the K-means algorithm for 6 clusters

correct done for nodes with 1, 2, 3, 4, 5, 6 and 7 processes while for nodes with 8 processes classification is not correctly done. Average absolute error using K-means algorithm is 0.1419491525423729.

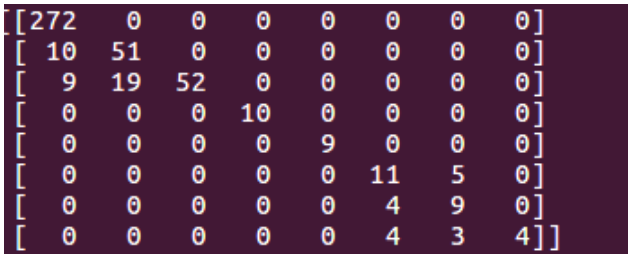


Figure 8. Confusion matrix for dynamic network coding using decision tree

Based on the results obtained by each of the machine learning algorithms, we can observe that the decision tree presents a better performance than K-means to predict the number of roles that a node can execute at the same time, because decision tree has an average absolute error of zero, and its confusion matrix shows no error. Likewise, the coordinating server each one that will initiate a network coding process, assigns roles based on the state in which each node is located depending on the RAM, CPU, disk and number of processes that it is carrying out, which has allowed that most processes are carried efficiently and that servers are not saturated, which has allowed the system to have better load balancing, streamline processes and take advantage of node resources.

V. Conclusion

Network coding has had a positive impact on modern communication networks, and we can found several application areas of network coding in the literature. In this work, we have evaluated the impact of combining dynamic network coding with techniques of machine learning. In this scenario all nodes in the networks can perform network coding. This vision introduces several benefits in a collaborative system because the nodes not only share files but also processing capacity in all participating nodes. Also there is a best load balancing related to processing. In this work results report that the best technique that allows the coordinator server to determine the most optimal way to use the resources of the peers using machine learning is the decision tree. Using decision trees, the coordinator server can assign in a more optimal way the roles of each peer, and the number of times a peer can take that role during dynamic network coding operations. In this case the decision tree presents a better performance than the K-means technique. Performing an optimal allocation of resources and roles in the system prevents the nodes from becoming saturated or falling due to overload during network coding operations. As future work we plan to implement algorithms to synchronize the matrices and pass all the matrices to distributed databases. The system can also be made more robust by implementing some security mechanisms.

References

[1] H. Pucha, G. d. Andersen, and M. Kaminsky. "Exploiting Similarity for Multi-Source Downloads Using File Handprints," 4th USENIX NSDI '07, Cambridge, MA, USA April 2007.

[2] F. A. López-Fuentes, and E. Steinbach, "Multi-source video multicast in peer-to-peer networks," Proc. of the IEEE International Symposium on Parallel and Distributed Processing, pp. 1- 8, Miami, FL, USA, 2008.

[3] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, 46, 1204–1216, 2000.

[4] J. Mendoza-Almanza, and F. A. López-Fuentes, "Collaborative Multi-source Scheme for Multimedia Content Distribution," WITCOM 2016, Silao, Gto. México, 2016.

[5] J. Mendoza-Almanza, F. A. López-Fuentes, R. Hasimoto-Beltran, "Practical Network Coding for Multi-source Scenarios," Smart Technology, Springer, pp. 141-148, 2018.

[6] L. Keller, E. Atsan, K. Argyraki and C. Fragouli, "SenseCode: Network Coding for Reliable Sensor Networks," *Tech. Rep. 2009, Ecole Polytechnique Federale Lausanne (EPFL).* Last updated July, 2010.

[7] P. Chou, Y. Wu and K. Jain, Practical network coding, *51st Allerton Conf. Communication, Control and Computation,* Monticello, IL, USA, 2003.

[8] C. Gkantsidis and P. R. Rodriguez, "Network Coding for Large Scale Content Distribution," *IEEE INFOCOM 2005,* Miami, FL, USA, 2005 2235–2245.

[9] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher and J. Barros, "Network coding meets TCP," IEEE INFOCOMM 2009, Rio de Janeiro, Brazil, 280–288, 2009.

[10] S. Surati, D. C. Jinwala and S. Garg, "A survey of simulators for P2P overlay networks with a case study of the P2P tree overlay using an event-driven simulator," Engineering Science and Technology, 20 (2017), pp. 705–720.

[11] T. Matsuda, T. Noguchi, and T. Takine, "Survey of network coding and its applications," IEICE Transactions on Communications, 94(3), 698–717, 2011.

[12] F. Jamil, A. Javaid, T. Umer, E. and M. H. Rehmani, "A comprehensive survey of network coding in vehicular ad-hoc networks," Wireless Networks, 23(8), 2017, pp. 2395–2414.

[13] D.Nguyen, C. Nguyen, T. Duong-Ba, H. Nguyen, A. Nguyen and T. Tran, "Joint network coding and machine learning for error-prone wireless broadcast," IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), 2017.

[14] Z. Yang, M. Li, and W. Lou, "A network coding approach to reliable broadcast in wireless mesh networks," Wireless algorithms, systems, and applications, 2009, pp. 234–243.

[15] L. Rokach and O. Maimon, "A Top-down induction of decision trees classifiers-a survey," IEEE Transactions on Systems, Man, and Cybernetics, Part C. 35 (4), pp. 476–487, 2005.

[16] A. K. Jain, "Data Clustering: 50 Years Beyond K-means," Pattern Recognition Letters 31 (8), pp. 651–666, 2010.

[17] F. A. López-Fuentes and J. Mendoza-Almanza, "Dynamic Network Coding for Collaborative Multisource System," IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2018.

Peer-to-Peer Networking and Applications (2019) 12:724–739
https://doi.org/10.1007/s12083-018-0668-7

Hierarchical P2P architecture for efficient content distribution

Rogelio Hasimoto-Beltran[1] · Francisco de Asís Lopez-Fuentes[2] · Misael Vera-Lopez[2]

Received: 31 August 2017 / Accepted: 6 July 2018 / Published online: 1 August 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract
Mutualcast is a one-to-many (peer-to-peer) scheme for content distribution that maximizes the overall throughput during a broadcast session. It is based on a fully-connected graph (full mesh topology), which introduces benefits such as robustness or simultaneous transmission from/to multiple devices. The main disadvantage of Mutualcast is scalability; it is constraint to a small P2P group for content distribution. In this paper, we make Mutualcast scalable. We propose a highly collaborative and scalable P2P tree-based architecture made of two main components: 1) Peer grouping or clustering and 2) Hierarchical tree-based content distribution. In step 1), peer nodes (content receivers) are grouped into equal-size clusters by using a proposed heuristic size-constrained algorithm based on k-means. In step 2), clusters (which become the nodes of the tree) are organized into a single hierarchical n-ary tree-based architecture, in which the root of the tree (Root Cluster) is the one closest to source peer, while intermediate and leaf clusters are positioned in the tree according to their delay-proximity to previously inserted clusters. During content distribution, the root cluster receives the blocks of content before any other cluster in the tree and directly from (and only from) the source peer; blocks are then passed on to the next hierarchical level down the tree in order (higher levels of the tree receive the content before lower levels). Inter-clusters and intra-clusters content distribution is performed concurrently and takes into account peers upload/download capacities to relay blocks of content. The evaluation of our hierarchical P2P architecture concentrates on the following metrics: scalability of the systems, overall end-to-end delay distribution, and efficient cluster size. Finally, our architecture is compared against two well-known P2P technologies in the literature, Super-Peer and Kademlia.

Keywords Peer-to-peer networks · Video multicast · Content distribution · Clustering

1 Introduction

Multimedia content distribution over the Internet has increased at a very fast rate with significant impact on today's global economy. Popular services such as videoconferencing and Internet Protocol Television (IPTV) markets are expected to reach USD 7.94 and 95.9 global Billion by the 2020s, respectively [1, 2]. These services make use of media multicast technologies where information is addressed to a group of destination computers simultaneously using one-to-one (unicast) or one-to-many (multicast) schemes. Research teams in academia and industry worldwide are making significant efforts to innovate multicast architectures to address the challenges of a rapidly increasing market.

✉ Francisco de Asís Lopez-Fuentes
flopez@correo.cua.uam.mx

[1] Centro de Investigación en Matemáticas, A.C, Guanajuato, Mexico

[2] Universidad Autónoma Metropolitana, Mexico City, Mexico

Traditionally, Internet Protocol Multicast (IPM) has been proposed as an efficient solution for one-to-many media dissemination [3]. IPM is more efficient than unicast due to its reduced transmission overhead from the sender to all receivers. IPM decreases traffic by simultaneously distributing a single copy of data packets to thousands of users through networks routers. However, IPM has not been fully deployed in the Internet due to network control and management issues raised by Internet Service Providers (ISP). Thus, the deployment of IP multicast is currently limited to local area networks, and a handful of ISPs networks [4]. To address these issues, researchers have proposed an application level solution as an alternative to implement IPM [5, 6]. For example, in Application Layer Multicast (ALM), all tasks are implemented by collaborative work in the end-users exclusively, while the network infrastructure is kept fixed. ALM approaches provide more flexibility and are easier to deploy than those requiring network router multicast support.

In addition to ALM technology, peer-to-peer (P2P) computing technology has emerged as a novel paradigm to face some of the limitations of the client-server model [7]. The end

users provide all the communication infrastructure needed, so dedicated infrastructure is not required. Each user provides a communication node, and all the nodes comprise a network abstraction on top of the physical network known as an overlay network, which is independent of the underlying hardware network implementation. In the P2P systems, each peer can take the role of both server and client at the same time, so there is no need for dedicated servers. Due to the sharing of peer resources, the ALM scheme is an effective means for conducting the cooperative P2P communications. Namely, during a multicast session, peers contribute their resources to relay the media to others. In this way, as a new peer arrives to the P2P system, the demand is increased, but the overall capacity also increases. This feature is not available in a system based on a client-server model. In a P2P multicast system, the media must be delivered to all requesting peers with high quality and minimal delay. An overlay P2P multicast does not require any router support and is the most flexibility and adaptable to diverse requirements from these applications.

Most P2P multicast implementation algorithms can be classified according to the data structure used to support packet distribution (i.e. trees, forests, or fully-connected graph) [5]. In conventional tree-based distribution algorithms, the peers placed as interior nodes redistribute data content, while the peers placed as leaf nodes only receive data. Although the multicast-tree based scheme is highly scalable [8–10], it is not maximally efficient in collaborative environments, because the upload capacity of the leaf peers is not used during a multicast session. The full upload capacity of all participating peers is required in order to achieve maximum throughput. A possible solution to increase efficiency consists on constructing multiple concurrent trees, where peers contribute with their upload capacity in at least one tree or in the construction of a fully-connected network. A drawback of using a fully-connected graph (or mesh architecture) is that the number of connections is proportional to the number of peers, because each peer has to forward its received blocks from source to all other peers. Mesh-based approaches also have high control overhead due to data scheduling and limitations for delay sensitive applications when the participating peers are located in different geographical locations. On the other hand, the dynamic behavior of peers in P2P systems is one of the major challenges. Since peers are transient in nature, once a parent peer departs from the multicast system [5], the receivers receiving streaming content from that parent peer might suffer a temporal interruption in the content transmission.

In this paper, we propose a fully collaborative and scalable P2P architecture which involves strong cooperation between participating peers during the content distribution from a source to multiple peers. Participating peers are organized into different clusters or groups based on delay-proximity. Peer delay-proximity is exploited in our proposed scheme in order to form a fully hierarchical cluster of peers interconnected via a single n-ary tree [11], with excellent content propagation time. The source-peer (root of the tree) divides the content into blocks and distributes different blocks to all peers in the highest hierarchical cluster (root cluster), so that each peer within the cluster contributes its redistribution capacity by forwarding the receiving blocks to the rest of peers within its own group and receiving at the same time the rest of blocks not directly obtained from the source peer. An n-subset of peers within the cluster is designated as source for the n lower clusters in the next set of clusters down the hierarchy tree structure, that is, one peer is designated as source for each receiving cluster. The process continues in the same way, until all cluster leaves are reached. We evaluate our proposed architecture based on the overall end-to-end delay distribution to all peers, tree-based scalability, and cluster size. A comparison against two well-known P2P technologies in the literature, such as Super-Peer [12] and Kademlia [13] is presented.

The remainder of this paper is organized as follows. We introduce and discuss some collaborative multicast approaches in Section 2. We briefly explain how to build the collection of clusters connected via a simple tree in our proposed architecture in Section 3. How the collaborative architecture is implemented in the simulator is explained in Section 4. In Section 5, we evaluate the performance of our collaborative architecture against other content distribution schemes. Section 6 concludes the paper.

2 Collaborative multicast schemes

In this section, we describe the main technologies our scheme is based on: mesh-based approaches (such as Mutualcast) and tree-based approaches. Mutualcast has shown to be a scheme that maximizes the overall throughput during a multicast session. In addition, Mutualcast is based on a fully-connected graph (full mesh topology), which introduce benefits such as robustness or simultaneous transmission from multiple devices. On the other hand, a tree-based scheme introduces several benefits such as scalability, reduced end-to end delay and easy maintenance. Our aim is to reach shorter end-to-end delivery time, improve scalability and low resources consumption by merging these two technologies into an efficient content distribution scheme.

2.1 Tree-based approaches

In a tree-based approach, an overlay construction mechanism organizes participating peers into a single tree whose root is located at the source node. The participating peers are organized into a single tree following their classification as interior node or leaf node. In a tree-based topology, the source peer sends the data to the requesting peers on the first level, which then forward the data to the requesting peers located on the

following level down the tree structure and so on until reaching the leaf peers. In this configuration, a video stream is pushed from a parent router to its children routers along a well-defined route. In this way, the multicast tree for content distribution uses the upload capacity of the peers located on the intermediate levels. However, the upload capacity of the leaf peers is not used. Although a tree approach probably represents the most effective distribution structure in terms of bandwidth and delay optimization [14], this configuration has an inherent drawback because all the burden generated by forwarding multicast messages is carried out by a relative small number of interior nodes.

2.2 Mesh-based approaches

In a mesh-based overlay, a peer can concurrently receive data from different senders, each contributing a portion of its upload capacity. Additionally, the requesting peers can send and also receive data from each other. Video data in a mesh-based P2P multicast is available in multiple neighboring peers, with a node having to pull data to avoid significant redundancies, while in a forest based overlay the data is pushed from a parent peer to many child peers. Due to the dynamic and unpredictable behavior of peers, the main challenge of a mesh-based overlay is how to select the proper senders [15] and how to cooperate and schedule the received data in the requesting peers. In a collaborative environment such as a P2P network, the participating peers contribute with resources proportional to the benefits they obtain from the system. Specifically, in an application layer multicast, the peers expect that the forwarding load will be shared among all participants [6]. However, a multicast based on a single tree does not match well with these cooperation expectations, because a small number of interior peers carries the forwarding multicast traffic, while the upload capacities of a large number of leaf peers are not used. This is a critical problem for applications with high bandwidth requirements such as video or bulk file distribution, because many interior nodes in the multicast tree may not have the required upload capacity. To face these challenges, our proposed scheme adopts a tree structure as the global structure but incorporates small mesh clusters on each level of our single distribution tree. Clusters are an elementary unit in this hierarchical architecture, which involves one source peer and several requesting peers. The peers inside a cluster are fully connected, and each peer inside a cluster is a receiving and forwarding peer at the same time. Due to the fact that the upload capacity of all peers is also used, the bandwidth consumption from the source can be reduced.

2.3 Hierarchical clustering approaches

Tree and mesh overlay topologies have been found not suitable for large scale dynamic P2P networks; they become inefficient and involve high control overhead. The concept of hierarchical clustering has emerged as a new alternative in which, peers are grouped into clusters and clusters into an organized tree topology. In NICE scheme [9], a balanced tree of clusters is built, in which all peers are part of the lowest layer including the source peer. Higher layers of the tree are represented by corresponding cluster centroids of lower layers, in this way the root of the tree is the centroid of all cluster centroids of lower layers. This model simplifies the insertion of peers in the hierarchical tree. NICE uses the head to forward the content to its subordinates, thus incurring a high bottleneck. Additionally, NICE tree-structure is fixed and not optimal; it does not provide the best low-latency distribution tree. Broadly speaking, it becomes a special case of our proposed hierarchical scheme. An extension of NICE is presented in [16], called Zigzag protocol. It is derived from the same balanced multicast tree developed in NICE, with a modified intra-cluster communication strategy. In this new strategy, intra-cluster peer communication is not allowed and each peer must relay completely to subordinate cluster or peers. Zigzag extends the nomenclature of the administrative organization of peers, claiming a reduced control overhead compared to NICE. One of the main drawbacks of Zigzag is peer-insertions, which occurs whenever there is place available in a cluster, affecting the transmission delay. A more recent scheme named TURINstream [17], combines a tree structured P2P video streaming scheme with Multiple Description Coding (MDC) to achieve low-delays, robustness to peer dynamics and limited protocol overhead. In MDC, video is composed by independent and complementary descriptions which can be decoded independently, yielding the base video quality (the more descriptions are received the better the quality of the video). The advantage of MDC is playback continuity despite peers´ departures, failures, and churning. The algorithm for building the tree is very simple, clusters must provide the upload capacity for a continuous transmission; it does not pay attention to optimal transmission delays efficiency. This is the main problem of TURINstream; a peer can be joined at any level of the network, it just needs to follow a path along the control tree until it finds a cluster that can host it (just based on the upload capacity).

Our scheme, is focused on improving the deficiencies of the above algorithms by building a new hierarchical tree topology that improves transmission efficiency in several ways: it reduces upload bandwidth usage, peer communication stress, and increases transmission robustness.

3 Proposed approach

Our underlying ground on proposing a new scalable scheme is that peers can greatly benefit from the capacity of other requesting peers via collaboration. Collaboration becomes a key factor for efficient multicast applications over large-scale heterogeneous environments. Based on these facts, we focus on developing a

collaborative computing system considering both the dynamic behavior and scalability of the networks. To achieve this goal, our proposed architecture is mainly constructed with a superposition of two overlapping networks, one using the tree model, which is the main structure (the body of the architecture), and the other using the mesh model (Mutualcast [18]).

3.1 Tree distribution model

Mesh-based P2P multicast (such as Mutualcast) can achieve the maximum overall throughput but incurs scalability limitations because all nodes are fully connected. To deal with these limitations, our proposed architecture uses clusters of peers allocated in a unique tree-rooted distribution at the source node. The hierarchical structure of our approach is shown in Fig. 1. The first level of the network hierarchy is the peer source (root node) that contains the original file. Initially, active peers in the system are grouped into small clusters (see section 3.2 for details), ensuring that peers closest to the root node (source peer S) will form the root cluster in the distribution tree (cluster 1). Peers with longer time proximity to the source peer are grouped as intermediate and leave clusters in the hierarchical tree. Leaves (cluster 2–4) have the longest time proximity to source peer. In this work, we consider clusters with the same number n of peers, but it can be easily extended to unequal clusters size (as a future work). In Fig. 1, peers P1, P2 and P3 form the cluster with the highest hierarchy level in the multicast session, while the rest of clusters and corresponding peers are subordinates. That is, information is first distributed from the source node to the root cluster, and from the root cluster to the rest of clusters following a sequential top-down order along the tree. Each peer forwards the blocks received from the source to the rest of the peers within the same cluster, and simultaneously receives the rest of the blocks from the other peers in the cluster. Peers in the same cluster share bidirectional communication. Concurrently, each peer in the first cluster acts as a source for a new cluster located on the second level of our hierarchical structure. Thus, peer P1 is a forwarding peer of cluster 1 and a source peer of cluster 2 (which is formed by peers P4, P5 and P6) at the same time. Peers P2 and P3 can also extend their own clusters. We denote cluster 2 as a child-cluster of peer P1.

The communication between peers located in the first cluster and the requesting peers clustering on a second level is unidirectional. In other words, in the hierarchical approach, the blocks are distributed from one cluster to another, in a top-down fashion. Using clustering, the peers can greatly benefit from the capacity of other neighboring requesting peers via local collaboration while the number of connections is reduced in comparison to a fully connected overlay topology. The total number of connections TC (for a constant cluster size) in our hierarchical scheme can be represented by:

$$TC = k*\left[\frac{(n)(n-1)}{2}\right] + l*n + \left[\frac{(p)(p-1)}{2}\right] \quad (1)$$

[internal external residual]connections

where $n = \lfloor N/k \rfloor$ represents the cluster size, N is the number of peers in the system, k is the number of clusters, l is the number of links in the tree (external node-to-node + source-to-root connections), and remaining peers $p = (N \bmod k)$ are allocated *in* a final *p-size* cluster. In a multicast group with $N = 150$ requesting peers, $k = 30$, $n = 5$, and $l = N$, our proposed architecture needs 1050 connections to distribute all the blocks. In contrast, using a fully-connected architecture (e.g. Mutualcast [18]), the overlay network is formed with $(N-1)(N-2)/2 = 11026$ connections. In this way, our proposed architecture is scalable and robust at the same time. The overall delay optimization problem for minimizing the content distribution time is more complex than just considering the number of connections. It involves N, n, peers´ upload and download capacity, and the final structure of the distribution tree. In the next and

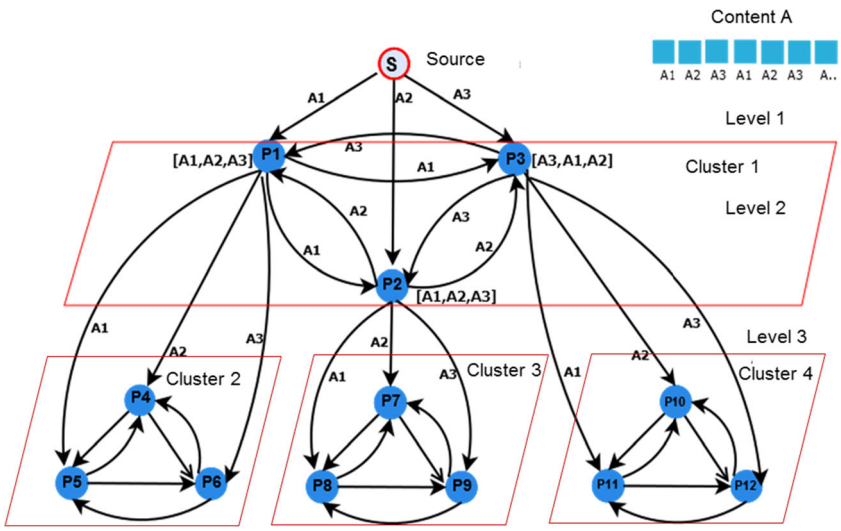


Fig. 1 Scalable collaborative multicast

experimental sections, we will address this problem in detail. For the moment, we concentrate on a very important step in our scheme, peer clustering.

3.2 Constraint clustering process

In this work, we use the Round-Trip Time (RTT) between two peers as proximity information to build the local clusters. Given a data set $RTT_i = \{rtt_{i,\ 1}, rtt_{i,\ 2}, \ldots, rtt_{i,\ n}\}$, $i = 1, \cdots, N$, representing the Round-Trip-Time (RTT) from the i^{th} peer to all active peers in the system, our aim is to partition the *N(N-1)* observations into k mutually exclusive clusters $S = \{S_1, S_2, \cdots, S_k\}$ that minimize the sum of squares (within the cluster) given by:

$$arg\ \min_{S} \sum_{j=1}^{k} \sum_{rtt_{l,m} \in S_j} \left(rtt_{l,m} - \mu_j \right)^2 \tag{2}$$

where μ_j is the centroid of the cluster S_j, whose cardinality is $|S_j|$. The solution of Eq.2 for a global minimum is an NP-Hard problem, since there exist $k^N/k!$ different ways for grouping an *N(N-1)* data set [19]. Instead, several heuristics have been developed to provide local minimums or suboptimal solution to this problem, the simplest and most widely known is the k-means or Lloyd's algorithm. Lloyd's algorithm is based on the simple observation that the optimal placement of a center is at the centroid of the associated cluster. The algorithm proceeds as follows [20]:

Algorithm 1: k-means

1. Select k random centroids for the initial partition of the data space.
2. Assign each data point *rtt* to the cluster corresponding to the closest centroid:
   a. For each cluster centroid μ_j, compute the distance between $rtt_{\mu_j,i},\ j = 1, \cdots, k;\ i = 1, \cdots, N^2$.
3. Calculate the new centroid of each cluster.
4. Repeat steps 2 and 3 until the algorithm converges (centroids do not change anymore).

The k-means baseline algorithm has been modified to satisfy the following constraints imposed in our hierarchical tree-base model: **a)** the centroid of a cluster must always be a peer; **b)** RTT values are nonsymmetrical; and **c)** the number of peers in the clusters must be small and constant. The first constraint avoids the use of fictitious peer centroids for which we cannot measure RTT distances from/to any peer (because of the non-linearity in the data). A major benefit of this constraint is faster convergence time since the algorithm does not need to re-compute the RTT values for each iteration as in the original k-means. The second constraint (complements the first constraint) takes into consideration that RTT is not symmetric, that is the distance from A to B is not necessarily the same as the distance from B to A. Since RTT values are 1-D scalars without intrinsic spatial distribution information, the way to recalculate the new centroid position of cluster S_l in step 3 of algorithm 1, is by finding the peer for which:

$$\min\left(\sum_{j=1}^{S_l} rtt_{i,j} \right), \forall i, j \in S_l \tag{3}$$

as depicted in Fig. 2. Let us consider that all peers in S_l can be spatially located as shown in Fig. 2a (this is not possible in the real world), and the distance from peer *i* to all peer *j's* can be schematically represented as shown in Fig. 2b. It is easy to see that the minimum of Eq. 3 corresponds to peer $i = 5$, which becomes the new centroid of the cluster. Figure 3 shows the final clustering (one instance of many possible) of the k-means for $N = 20$ peers, $k = 4$ clusters and $n = 5$. Figure 3a shows the initial peer distribution and Fig. 3b the corresponding clustering output.

The last constraint mentioned above (constraint c), is related to Mutualcast mesh connection limits, which is approximately <15 nodes. There exist good solutions to this problem in the literature (see [21]) with increased time and implementation complexity (requires the use of linear programming). Instead, we developed a simple heuristic approach to satisfy the cluster size constraint once k-means is applied (our scheme takes advantage of the sub-optimal k-means output). It is worth mentioning that more than finding an optimal partition of peers, our main contribution is the hierarchical approach, in which information is being transmitted in both ways, horizontally and hierarchically vertical at the same time. Given the output of algorithm-1 and cluster size $|S_j| = n = N/k, j = 1, \ldots k$, our size-*constrained algorithm* consists of the following steps (If $n = N/k$ is not integer, one of the cluster will have $n + (N \bmod k)$ peers):

Algorithm 2: Cluster size constraint

Input Data: k-means clustering (Algorithm-1)
Output Data: n-constrained peer clustering

1. Create a Cluster Distance Array CDA in ascending order using SP as a point of reference (closest clusters to SP are on top of the array).
2. For each cluster Sj , j = 1, . . . , k in CDA:
   a) If |Sj | = n, Sj is done and not considered for further peer exchange process.
   b) If |Sj | >n and j = 1, hand over the n − |Sj | farthest peers Pm (with respect to SP) to the closest cluster Si . Sj is done and not considered member of CDA for further peer exchange process. Update the centroid of S_j and S_i.
   c) If $|S_j| > n$ *and* $j > 1$, pick the closest peers to the centroid (it may include peers from other surrounding clusters) such that $d(P_s, \mu_j)$ is minimum, and hand over the $n - |S_j|$ peers P_h to the closest clusters, such that $d(P_h, \mu_i), m = 1, n - |S_j|; \forall \mu_i \in CDA$ is also a minimum. *d(X,Y)* is the RTT distance from *X* to *Y* and P_s are the peers staying in S_j. S_j is done and not considered member of CDA for further peer exchange process. Update the centroids of all modified clusters.
   d) If $|S_j| < n$, take $|S_j| - n$ peers from the closest cluster S_i, such that the distance $d(P_m, \mu_i), m = 1, j; \forall \mu_i \in CDA$ is a minimum. S_j is done and not considered member of CDA for further peer exchange process.

Generally speaking, algorithm-2 groups peers with minimum distance to the centroid. The exception is the closest cluster to source peer SP, for which it takes those peers closest to SP without considering their distance to the centroid. Figure 4, shows the output of algorithm-2 for $N = 20$, $n = 5$, and $k = 4$.

Clustering process is a centralized process (run by source node or a dedicate server). However, we have also considered its implementation in a distributed system where source node (or a dedicated server) and cluster centroids participate in the process. Arriving peers receive from source node (or dedicated server) an ordered list of all cluster centroids they may join in; peers select its closest cluster centroid and send a join-in request. If the cluster is full, it hands over its farthest peer (including the new peer in the computation) to another cluster, and receiving cluster repeats the peer insertion algorithm. When cluster centroid changes after a peer insertion, parent and children clusters are informed of the new changes (this is important for the forwarding content and control parameters). This

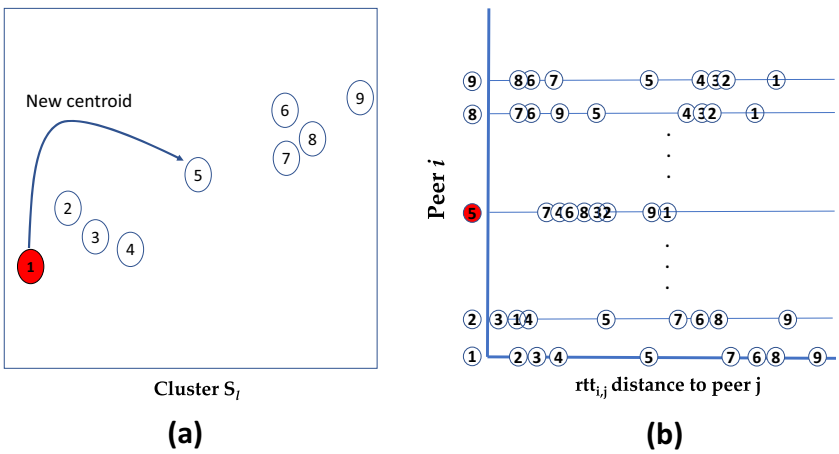


Fig. 2 Locating the new centroid from RTT measures. (a) Original Cluster, and (b) Representation of RTT distance from each peer i to all peers

distributed algorithm is highly scalable (the number of comparisons is at the most half of the number of centroids) and dynamically adapts (for optimal delay) on every single peer insertion.

3.3 N-ary tree creation

In our *n-ary* tree creation process, we favor (if possible) shorter communication links between peers and clusters for robustness, in particular for TCP connections. The tree is created in a top-down fashion with two pre-inserted nodes: the root of the tree or Root Cluster (RC), which is the one closest to the source peer (SP), and the first left child of RC, representing the closest cluster to RC. Hereafter, tree node insertions depend on the relative distance to their closest node and corresponding parent. Since higher hierarchical nodes send data to children, the RTT values considered for the insertion to the tree are from current nodes distance in the tree, to prospect nodes (already computed for the clustering process described in previous section). Let S_j be the next node to be inserted in the tree coming from a predefined node list (LC) ordered from closest to farthest distance with respect to RC, S_i be its closest node (already in the tree), and P the parent of S_i. If $d(P, S_j) < [d(P, S_i) + d(S_i, S_j)]/K$, then S_j becomes a child of P (or sibling of S_i) (if the number of children of P is less than n), otherwise is inserted as child of S_i. A special case during the insertion of cluster S_j is when P is RC; RC could take more than one child if RC has enough upload capacity. This is useful for reducing transmission delay when two clusters are close to RC but in opposite sides. Depending on the value of the constant K, it favors the creation of more balanced trees ($K > 1$), deeper trees ($K < 1$), or no influence at all in the final tree organization ($K = 1$). The example in Fig. 5 shows the following information: centroid distances of the clusters, clusters´ ordered list LC, and S_2, the next cluster to be processed (Fig. 5a). Since $d(RC, S_2) = 5 < 7 = [(d(P, S_i) = 3) + (d(S_i, S_j) = 4)]$ for $K = 1$, S_2 is linked to RC (parent of S_1) creating at this point a 2-level tree (root and two children). For $K = 1.5$, S_2 becomes a child of S_1, creating a 3-level tree, one node per level. The last iteration produces the final 5-ary tree shown in Fig. 5b and c for $K = 1$ and $K = 1.5$ respectively.

The tree creation algorithm is described as follows:

Algorithm 3: *n-ary* tree creation

Input Data: n-constraint k-means clustering (Algorithm-2)

Output Data: Content distribution tree

1. Get the closest distance cluster from SP, it becomes the Root Cluster (RC).
2. Get an ordered distance list LC (closest to farthest) between RC and the rest of clusters. The first cluster in the list becomes RC left child. Clusters will be inserted in the tree following the order of LC.
3. Compute the distance from all nodes in the tree to S_j, the next cluster in the ordered list LC. Let S_i (in the tree) be the closest node to S_j.
   a. If S_i is a child (has a parent P), compute $d(P, S_j)$, $d(P, S_i)$, $d(S_i, S_j)$ and make the following decision:
      i. if $d(P, S_j) < [d(P, S_i) + d(S_i, S_j)]/K$, and $nc < n$, S_j becomes a child of current parent (or sibling of S_i); otherwise S_i becomes a child of S_i.
4. Repeat step 3 until all nodes in LC have been inserted in the tree.

As part of the control topology, every peer manages a peer list with the following information: peer follower, peer source, and peer consumer. Followers are members of the same cluster who alertly watch a predefined partner; in the case of fail, the follower will take over its duties. Figure 6 shows an example of the information or control list carried out by all peers. Peers in the current cluster column watch for themselves, in this case, 1 is the follower of 2, 2 of 3, 3 of 4, and 4 of 1; if peer 3 fails or leaves, peer 2 takes over its functions including receiving from peer 6 (source) and forwarding to peer 5 (consumer). Peer 2 now watch for peer 4 in current cluster. Peers periodically sends keep-alive packets (acks every n packets) to its clustermates. When a peer fails or leaves, the cluster centroid initiates a peer request (in order to maintain the same number of peers) to children clusters. After the handover, the child cluster repeats the same action with its children until a leaf is reached. If current cluster is a leaf, it will stay as is. All peers maintain the last packet correctly received, so when

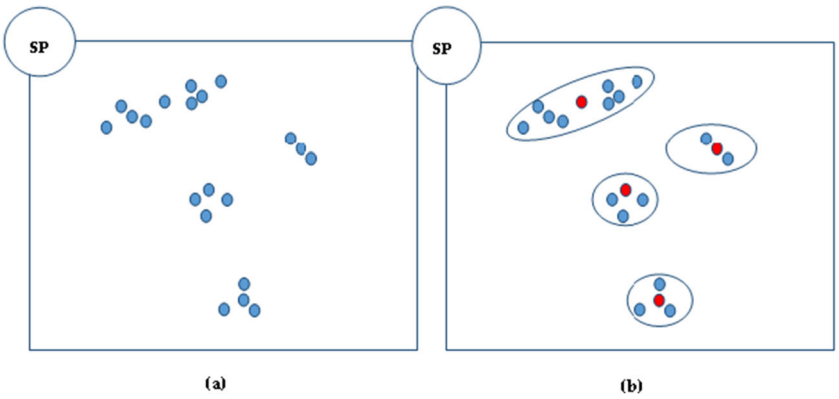


Fig. 3 (a) Original peer distribution, (b) k-means output (centroids are marked in red and SP is the Source Peer)

resuming transmission because of a peer fail, the algorithm ensures that all content will be received.

3.4 Intra-cluster and inter-cluster communication

Once the tree of clusters is created, the source node divides the content (e.g. a file or media stream) into small blocks, to be sent out to the highest hierarchical cluster or root cluster in the tree. Within the root, every peer is designated as receiver from the source node and as sender to the next and closest hierarchical level of clusters (as shown in Fig. 1). Every peer receives different blocks, which are concurrently redistributed to all peers within the cluster (*intra-cluster communication*) and at the same time to the next level of clusters down the tree hierarchy (*inter-cluster communication*). The same process is performed by lower cluster levels until source content reaches all peers in the leave clusters.

Similar to Mutualcast [18], an optimal bandwidth allocation strategy is implemented using redistribution queues between the source and requesting peers. In each cluster, a fully connected topology is built considering proximity information. Within each cluster, content is distributed among all participating peers, which are also called *requesting peers*. Peers are in fact receivers (Re) and senders (Se) at the same time. Each source splits the original content into small blocks and one unique peer is selected to distribute a block to the rest of the peers. Each requesting peer forwards the blocks directly received from a source to the rest of the participating peers in its own cluster. Peers with different upload capacity distribute a different amount of content. When the source peers have abundant upload resources, each source additionally sends one block directly to the receiving peers. Source sends one block to each participating peer for redistribution, one block in parallel to all requesting peers. Each requesting peer forwards the blocks received from the sources to the other requesting peers. After this, each peer works as a source for its own cluster. Each cluster is formed by the source S of upload capacity B_S and N_1 requesting peers R_i with an average upload capacity C_R. Each source S distributes its contents in two different routes: (1) through the content-requesting peers and (2) directly from the source. The route 2 is chosen only when the source still has upload capacity after exhausting routes 1. Thus, the distribution throughput Θ, which represents the amount of content sent to the requesting peers per second is defined as

$$\theta = \begin{cases} B_s, & B_s \leq B_R \\ B_R + \dfrac{B_S - B_R}{N_1}, & B_S \leq B_R \end{cases} \tag{4}$$

where

$$B_R = \frac{N_1}{N_1 - 1} C_R \tag{5}$$

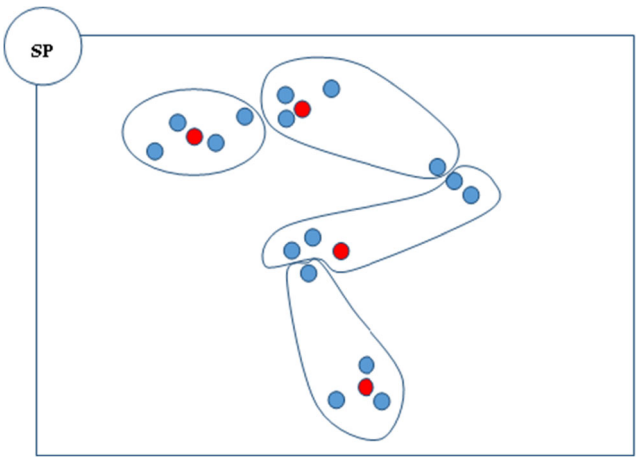


Fig. 4 Cluster size constraint

4 Implementation

This work adds scalability to the collaborative architecture presented in [11] and compares its performance with other similar architectures in the literature. To reach this objective we make use of a scalable P2P simulator called PeerSim [22]. This simulator is an extremely scalable simulation environment that supports dynamic scenarios such as churn and other failure models [22]. PeerSim has been written in Java

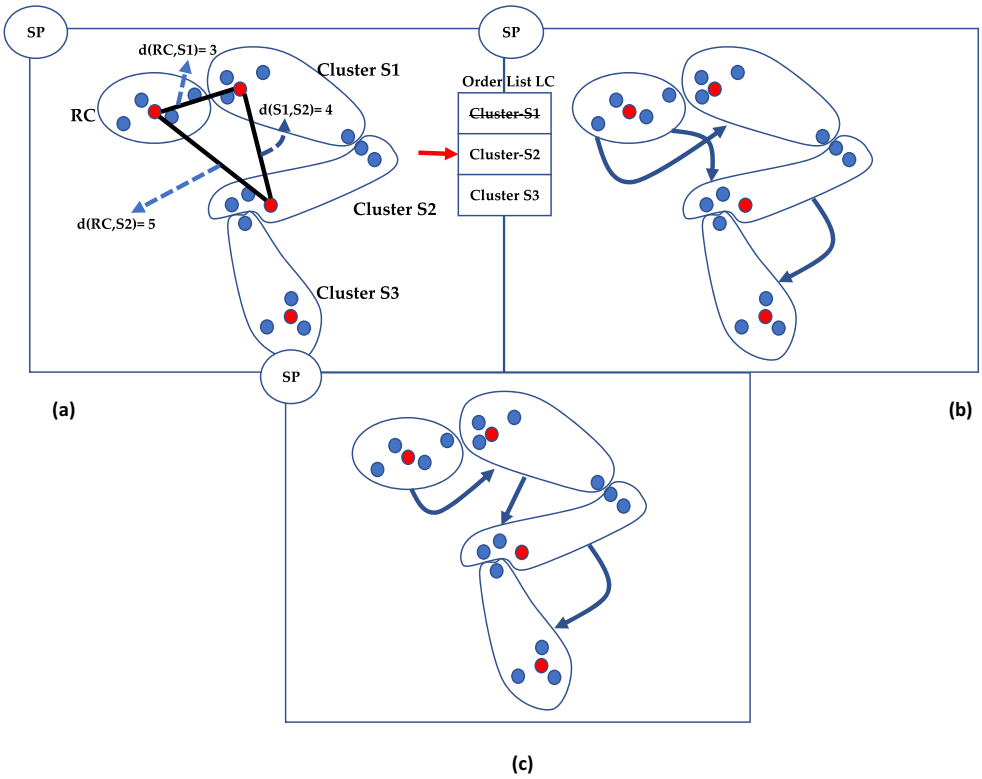


Fig. 5 Cluster size constraint. 5-ary tree creation process. (a) Initial conditions; (b) final tree for K = 1; and (c) final tree for K = 1.5

programming language, and the simulator classes can be extended to implement new peer-to-peer protocols. PeerSim consists of two simulation engines: cycle-based and event-driven. In cycle-based mode, authors claim simulation may reach 10^6 nodes. The engines are backed by many flexible components with a configuration mechanism, which can be fully configured and customized. The event-based engine is less efficient in terms of computing resources, but more realistic in its approach.

The PeerSim simulator is based on several components, which can be divided into protocols, nodes and controls. In order to improve the work environment, we use the Eclipse IDE due to its portability. To implement our collaborative architecture in PeerSim, we have developed a protocol called Hybrid Kademlia Protocol, which is a substrate between the application layer and the transport layer. Some classes from the Kademlia module [23] are taken as references and adapted in order to implement our protocol. Adapted modules of Kademlia protocol in Peersim are shown in Fig. 7.

The simulation module is customizable through a simple context using configuration files. These allow us to manipulate the parameters of all networks in order to establish the

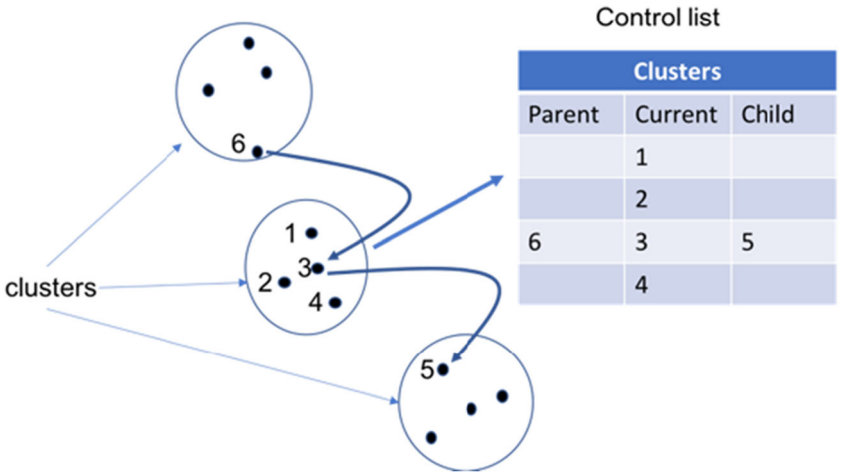


Fig. 6 An example of the control list carried out by all peers

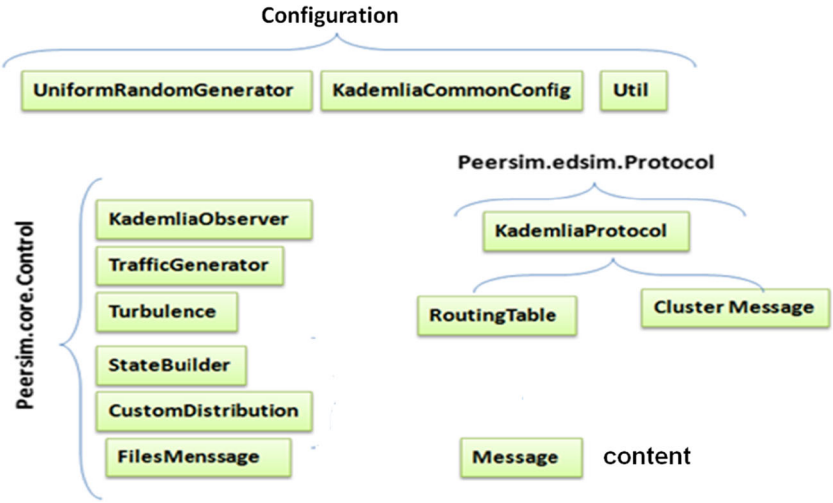


Fig. 7 Adapted modules of Kademlia protocol in PeerSim

various controls. Some of the parameters are BITS, which specifies the length of the ID, K, which is the number of calls to all replication system parameters, and ALPHA, which is the number of simultaneous search actions allowed by the protocol. The configuration file can invoke the command line to file. TXT, or can do so using the Eclipse IDE. The control protocol is very important because it allows us to simulate the dynamism and real scenarios of the nodes. These controls allow us to manipulate the traffic and the turbulence in the network, along with other events. Each control module allows for the identification of the peers who are outside the network or of the waiting time required for the distribution of content. Many of these controls have been pre-designed for the simulator. During the implementation of our architecture, a class known as Cluster class was created, which generates the group of nodes, distributes the content, and identifies the peer with its corresponding fragments. This class also allows for the manipulations of nodes within the hierarchical structure of our collaborative architecture.

Message routing plays an important role in our protocol, because when a message arrives at a node, the node can decide which route to use to send the message (a new node or the nearest node). Other classes created in our protocol are used to simulate the node's dynamicity, fragmentation of the content and the packet loss.

5 Results and discussions

We evaluate the scalability, intrinsic robustness, and cluster size of our proposed architecture based on the content distribution time to all peers using real and simulated experiments. Our first experiment, compares the performance of Mutualcast and the hierarchical collaborative multicast scheme using a small prototype over the PlanetLab Network [24]. Due to limitations of real experiments regarding the number of participant peers, we additionally simulated and evaluated the performance of our architecture in a second experiment using 500 and 1000 peers. Our third and final experiment, compares our architecture against

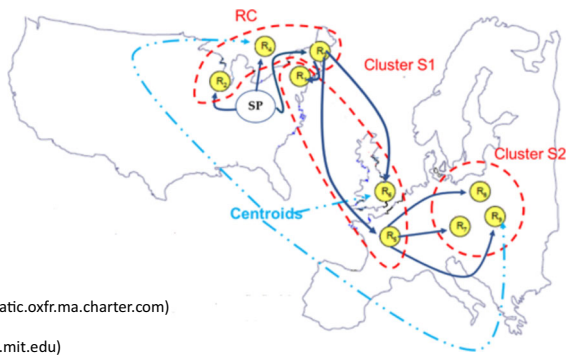


Fig. 8 PlanetLab experimental set-up and content distribution tree

PlanetLab nodes:
Source peer (SP):
SP – University of Pittsburgh (planetlab2.cs.pitt.edu)
Requesting peers:
R_1 – Worcester Polytechnic Institute (WPI): (75-130-96-13.static.oxfr.ma.charter.com)
R_2 – University of Chicago (planetlab3.cs.uchicago.edu)
R_3 – Massachussets Institute of Technology (planetlab7.csail.mit.edu)
R_4 – University of Toronto (pl2.csl.utoronto.ca)
R_5 – LIP6 – Université Pierre et Marie Curie (planetlab-01.lip6.fr)
R_6 – University College London – UCL (planetlab1.net.research.org.uk)
R_7 – Wroclaw University of Technology (planetlab1.ci.pwr.wroc.pl)
R_8 – TP-RD-Warsaw (planetlab1.warsaw.rd.tp.pl)
R_9 – Warsaw University of Technology (planetlab3.mini.pw.edu.pl)

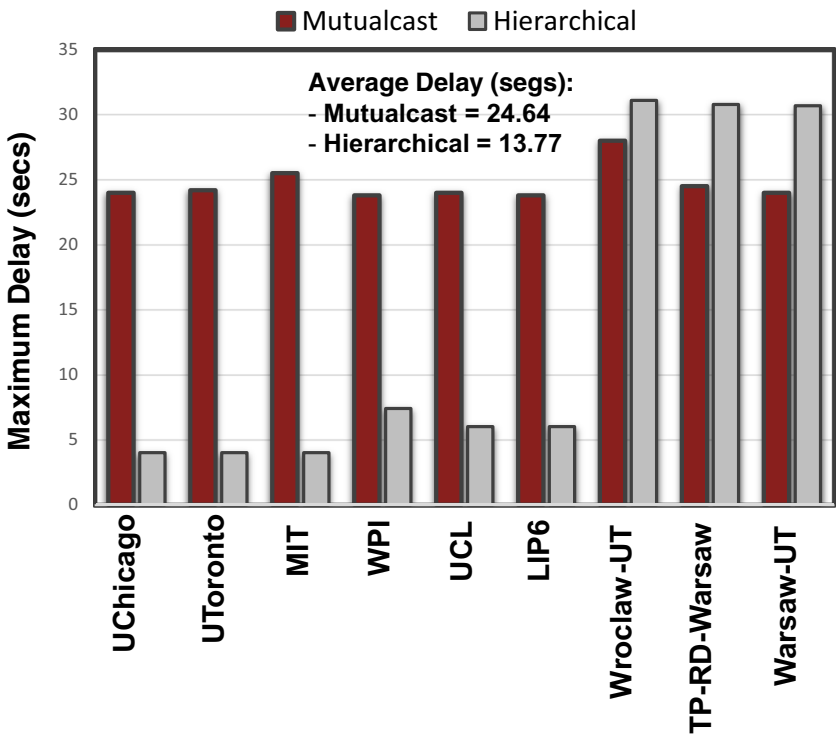


Fig. 9 Maximum delay comparison between Mutualcast and Hierarchical Collaborative [24]

two well-known P2P technologies in the literature, Super-Peer and Kademlia. In the experiments, 1.5 Mb and 5 MB files were distributed among all requesting peers.

5.1 Scalability: Hierarchical tree vs Mutualcast

A limitation of Mutulcast is related to its scalability. Although, part of our proposed architecture is inspired in Mutualcast, scalability is considerably improved by using clusters of peers organized into a unique distribution tree, that improves (in the average) the content distribution time. Scalability between our scheme and Mutualcast is compared in terms of delivery delay [24]. For this, a broadcast group of 10-peer PlanetLab [25] topology was created as depicted in Fig. 8. The SP is located at the University of Pittsburg, while requesting peers were spread out in the following academic centers: University College London (UCL), Worcester Polytechnic Institute (WPI), LIP6 (UPMC), MIT, University of Toronto,

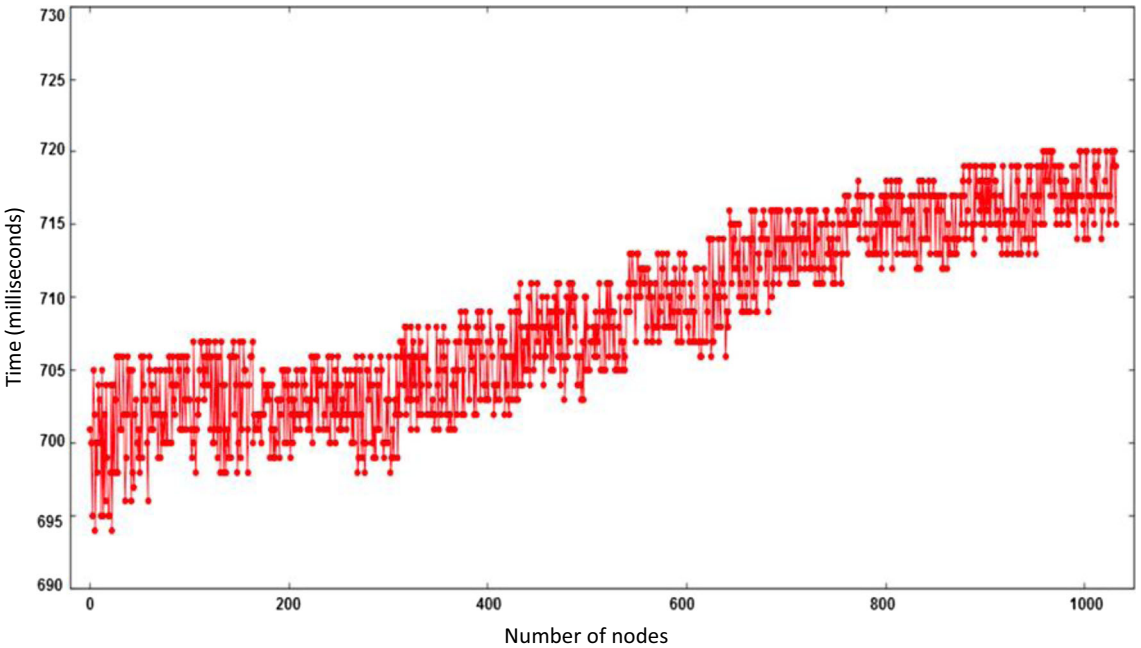


Fig. 10 PeerSim Propagation of a message in our collaborative architecture

University of Chicago, Warsaw-UT, Wroclaw-UT, and TP-RD-Warsaw. The resulting hierarchical tree after applying algorithm 1–3 (see section 3) is unary. That is, SP sends the blocks of content to closest peers R_2, R_3, and R_4 in RC; RC forwards the receiving blocks to cluster S1 (R_1, R_5, and R_6) through R_2 (the closest peer to S1 centroid), and finally, S1 forwards to S2 (R_7, R_8, and R_9) through R_5 (the closest peer to S2 centroid). During the clustering process, WPI peer was assigned to cluster S1 because of its PlanetLab connectivity was too slow despite the fact that spatially speaking it is closer to SP. Similarly, the connectivity of all nodes in Poland were also too slow, reason for which they were grouped by our clustering algorithm as the last cluster (S2).

Our Hierarchical Collaborative Topology (in Fig. 8) and Mutualcast were compared in terms of content delivery delay, in which thirty independent experiments were conducted over different days and times. The time delay in receiving the complete file content (of size 1.5 MB) at each peer was recorded,

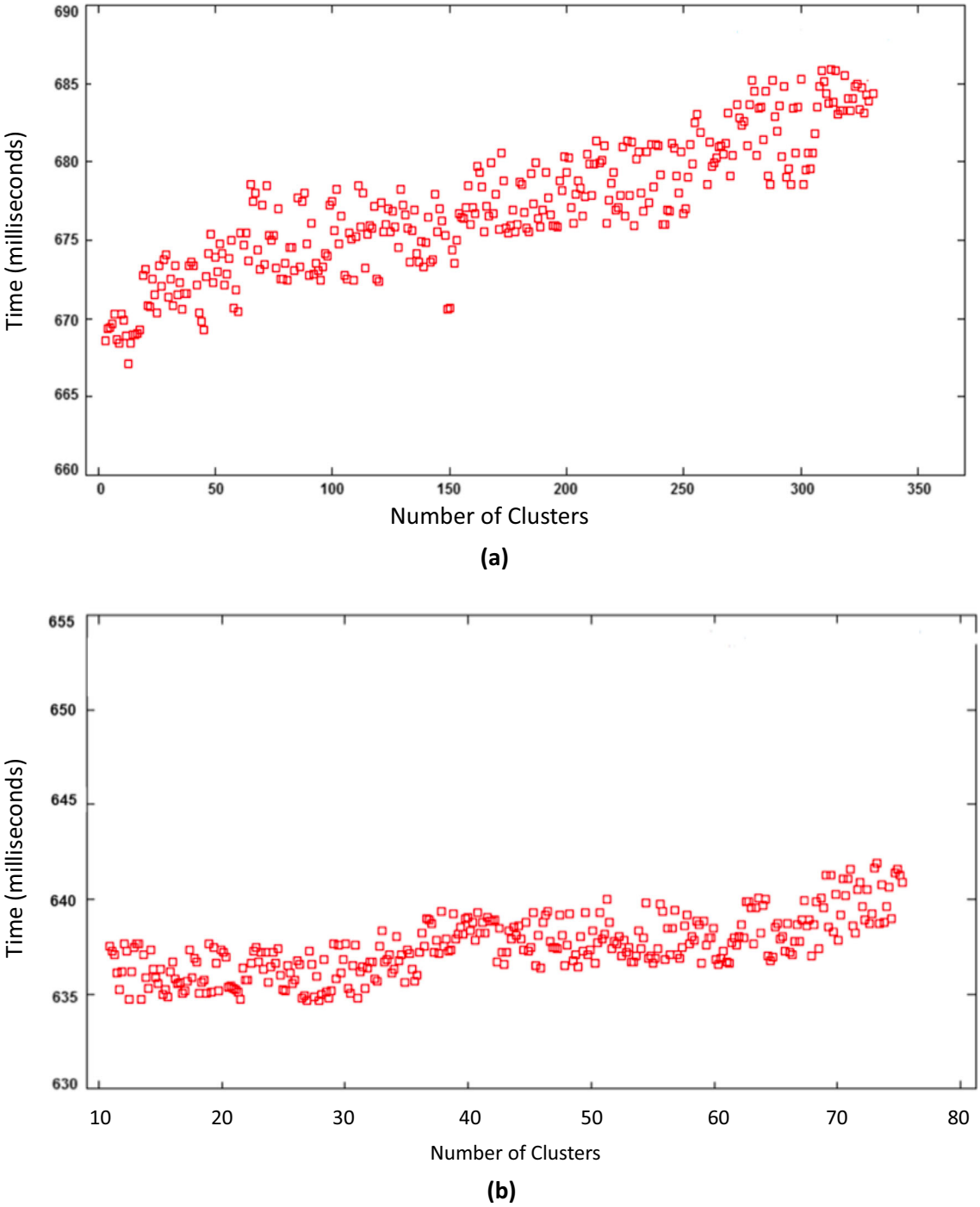


Fig. 11 Delivery delay. a) cluster size = 3 and b) cluster size = 12

and the scheme with smallest average delivery delay is assumed to present the best overall performance, as depicted in Fig. 9. When our approach is used, the average peer delivery delay is reduced by around 56% with respect to Mutualcast. This improvement is attributed to the fact that the source is close to the cluster at the highest level of the hierarchy. Thus, the throughput between the source and this subset of requesting peers is larger than the throughput between the source and the rest of the requesting peers (peers with slow connectivity are sent far down in the hierarchal tree to avoid affecting the overall content delivery delay per peer). The second fact is that using a hierarchical approach, the peers in the local clusters avoid the connection to distant peers into the overlay topology as in Mutualcast, wherein all requesting peers are fully connected. Slow peers in fully connected topologies (due to traffic, slow bandwidth, etc.), increase the average content delivery delay per peer (system is as fast as its slowest ink), as in the case of Poland, where all peers had bad connectivity.

In order to test the scalability of our scheme at higher levels with hundreds of peers, we simulated (see section 4 for details) the propagation delay of a 5 Mb file to all nodes in the *n-ary* tree network. The experimental set up consisted of $N = 1000$ peers, k = 333 clusters, cluster size $n = 3$ yielding a 10-level binary tree after applying algorithms 1–3 in section 3 (Mutualcast cannot handle this number of peers). Figure 10 shows results for this experiment, in which the first (1) and last peer (1000) in the horizontal axis are the closest and farthest to SP respectively. Our measurements of time (in milliseconds) were taken from the construction of the message until the reconstruction of the content and its defragmentation to generate a new hierarchical cluster level. Requests from peers building the cluster were intermediate operations in the protocol. Results show that the time spread of the content into the clusters increased consistently; as the tree architecture becomes deeper (increasing number of levels), intermediate and end peers require more time to regroup the total content of the transmitted media. The first five hundred peers receive the full content of the file more quickly than the rest of the peers, as expected. However, for this simulation the difference is not significant, while the first node received the complete content in 694 ms, the last node received it in 720 ms (all peers in the simulation are considered to have good connectivity and small RTTs).

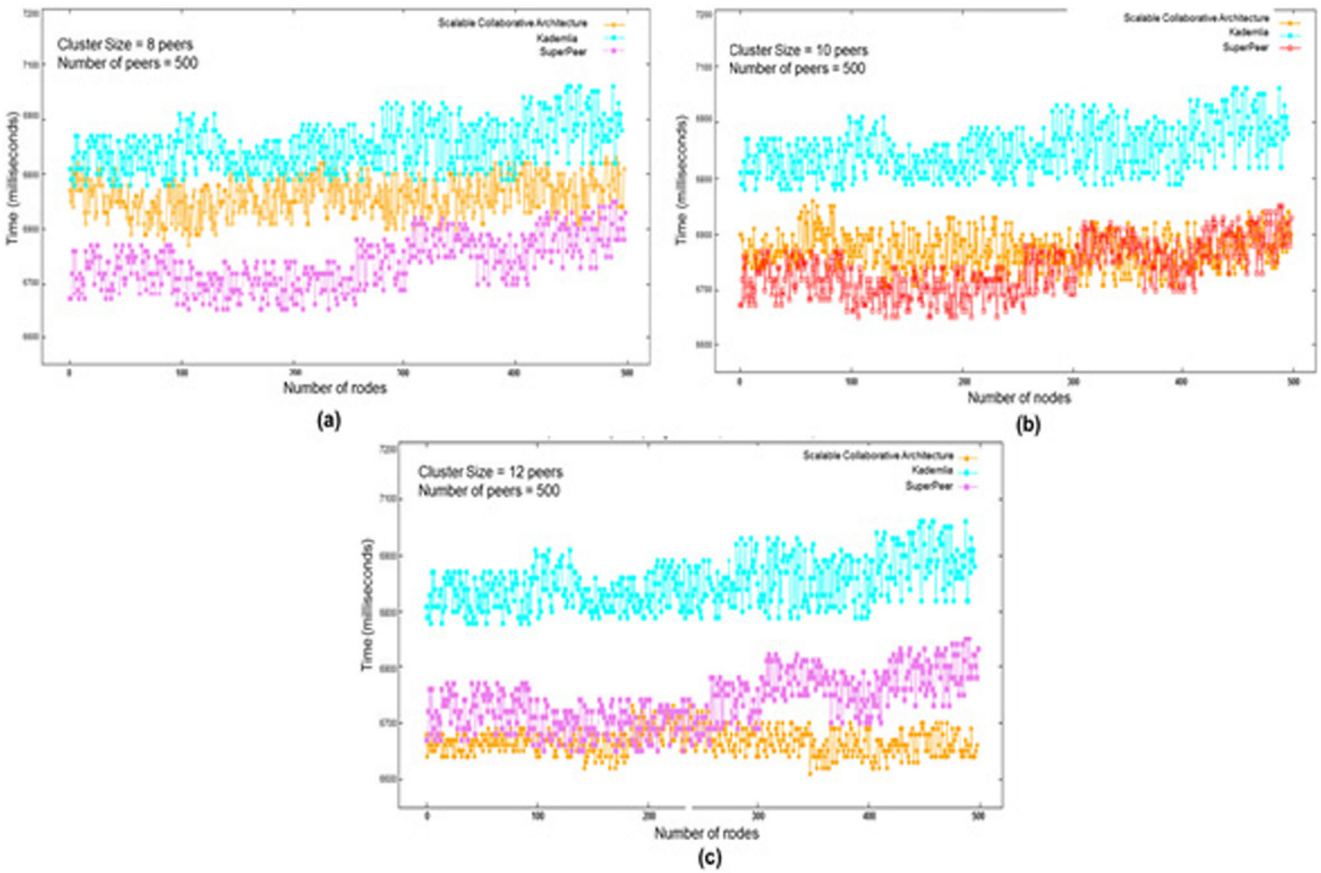


Fig. 12 Delivery delay comparison of Kademlia, Super-Peer and our Hierarchical (scalable) collaborative architecture. (a) cluster size = 8; (b) cluster size = 8; and (c) cluster size = 12

5.2 Cluster size impact

In this section, the impact of the size of the clusters during content distribution is evaluated using clusters of size 3 and 12 to distribute the content to 1000 peers, as shown in Fig. 11. These results represent the average measurements of ten simulations done with our proposed architecture. A collaborative architecture with clusters of size 3 requires a *n-ary* tree with 333 clusters to distribute the complete content to all peers, while an architecture with cluster of size 12 requires a *n-ary* tree with 42 clusters. Cluster with 3 peers commonly generates a deeper tree than cluster with 12 peers, causing a longer delay in distributing the full content to all nodes. In this case, the first peer received its complete content in 666 ms, while the maximum delay in which the last peer received its complete content was 686 ms. Clusters with size 12 had a better delivery time than the size 3 cluster; the first peer received its entire content in 634 ms, while the maximum delay to receive the complete content for all peers was 642 ms.

The results from our simulations show that the size of the cluster plays an important role in our architecture, because it introduces benefits in two important ways. The first benefit is derived from the facts that having a larger cluster means that there is a greater robustness and fault tolerance in the group. The second benefit is that the architecture gains scalability, which means that most of the requesting peers obtain the content more quickly. Traffic and turbulence factors allow us to generate a more realistic simulation of the network behavior. If a node loses communication with other nodes, it does not interfere with the reunification of the content because there are more nodes in the cluster that would provide support. For smaller clusters, a loss of content is more likely and peers have to retrieve the contents from a higher level of the distribution tree, which introduces a bigger delay. Architectures with small cluster (e.g. with 3 peers) are not robust because the clusters are very small, and if a peer left the cluster, it would be inoperative. Also, this type of architecture generates many small clusters, and the distribution tree requires many levels to organize all these clusters.

5.3 Comparison with other technologies

Finally, we compared the performance of our collaborative infrastructure with Super-Peer [12] and Kademlia [13]. We selected these protocols because of their similarities to our architecture proposed in this paper. Our simulation used a network with 500 nodes to evaluate these three architectures. In our first test, our collaborative architecture was constructed using a size 8 cluster. Figure 12a shows these results. In this case, we can see that Kademlia presents the highest distribution delay, while Super-Peer has the lowest distribution delay. The distribution delay of our architecture is between that of Kademlia and Super-Peer. Our second test considers a size 10 cluster in the collaborative architecture. The results from this experiment are shown in Fig. 12b. We can see how Kademlia continues with the same behavior, because its delivery times are high. However, delivery delay in our collaborative architecture is now very similar to the delivery delay in the Super-Peer architecture. In our last test, the cluster size in our collaborative architecture is increased to 12 peers as shown in Fig. 12c. In this case, our scalable collaborative architecture presents better performance than Kademlia and Super-Peer in terms of delivery delay, mainly in the farthest nodes (node 250 to node 500). Our results demonstrate that cluster size has an important impact on our scalable collaborative architecture, the bigger the cluster size the better the overall delivery delay. Since internally our clusters work as in Mutualcast, the cluster size is restricted to at the most 15 peers.

6 Conclusions and future work

In this work, we have developed a new hierarchical and scalable P2P architecture for fast and robust content distribution from a source to multiple nodes. In our architecture, we use time-proximity for grouping peers into clusters and clusters into a hierarchical interconnected *n-ary* tree in which, content is distributed concurrently within clusters (horizontal distribution) and among clusters in a top-bottom direction (vertical distribution). In the first place, we concentrated on evaluating critical issues in delay sensitive scalable computing systems, such as scalability (as a number of receiving peers and cluster size), robustness and delivery delay in our architecture. We found that our scheme performance (scalability and robustness) is proportional to cluster size. That is, as the number of receiving peers in a cluster increases the better the content distribution time and robustness of the system. In the second place, we compare our scheme against popular distribution schemes in the literature such as Kademlia and Super-Peer. Results show that our scheme provides a lower delivery time and better scalability, maintaining a reduced number of connections. As a future work, we are working on replacing Mutualcast content delivery in our clusters by an efficient optimized multicast scheme supporting a greater number of peers per cluster, capable of more demanding data content delivery such as video streaming.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Research Nester. Enterprise Video Conferencing Market: Global Demand Analysis Opportunity Outlook 2023. URL: http://www.researchnester.com/reports/enterprisevideo-conferencing-market-global-demand-analysis-opportunity-outlook-2023/164 (accessed on 01/05/2017)
2. Zion B. (2017) Market Research. IPTV Market for Advertising and Marketing, Media and Entertainment, Gaming, E-Commerce, Healthcare and Medical, Telecommunication It and Others - Global Industry Perspective, Comprehensive Analysis, and Forecast, 2015 2021. URL: https://www.zionmarketresearch.com/. Consulted May 1
3. Deering SE (1988) Multicast Routing in Internetwork and Extended LANs. In: Proc. of the ACM SIGCOMM, pp. 55–64, Stanford, CA, USA
4. Zhang B, Wang W, Jamin S, Massey D, Zhang L (2006) Universal IP Multicast Delivery, In: Computer Networks, Volume 50, 781–806
5. Gau V, Wu P-J, Wang Y-H, Hwang J-N (2002) Chapter1: Peer-to-Peer Streaming Systems. In: Ubiquitous Multimedia Computing edited by Q. Li, T. K. Shih. CRC Press
6. Chu Y, Rao SG, Seshan S, Zhang H (2002) A Case for End System Multicast. In: IEEE Journal on Selected Areas in Communications, Volume 20, Num. 8, pp. 1456–1471
7. Milojicic DS, Kalogeraki V, Lukose R, Nagaraja K, Pruyne J, Richard B, Rollins S, Xu Z. (2003) Peer-to-peer computing. In: Technical Report HPL-2002-57R1, HP Laboratories, Palo Alto, USA
8. Jannotti J, Gifford DK, Johnson KL, Kaashoek MF, Otoole Jr. J. W. (2000) Overcast: reliable multicasting with an overlay network. In: Proc. of the OSDI00, pp. 197–212, San Diego, CA, USA
9. Banerjee B, Bhattacharjee B, Kommareddy C (2002) Scalable Application Layer Multicast. In: Proc. of ACM SIGCOMM, pp. 205–217, Pittsburgh, PA, USA
10. Castro M, Druschel P, Kermarrec A-M, Rowstron A (2002) SCRIBE: A large-scale and decentralized application-level multicast infrastructure. In: IEEE Journal on Selected Areas in Communications,Volume 20, Num. 8, 1489–1499
11. López-Fuentes FA, Steinbach E (2007) Hierarchical Collaborative Multicast. In: Proc. of the 15th ACM Int. Conf. on Multimedia, pp. 763–766, Augsburg, Germany
12. Yang B, Garcia-Molina H. Designing a super-peer network. In: Technical Report, Stanford University. Available online: http://ilpubs.stanford.edu:8090/594/1/2003-33.pdf (accessed on 07/05/2017)
13. Maymounkov P, Mazieres D (2002) Kademlia: A Peer-to-peer Information System based on the xor Metric. In: Proc. of the Int. Workshop on Peer-to-Peer Systems, Cambridge, MA, USA
14. Wang F, Xiong Y, Liu J (2007) mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast. In: Proc. of the 27th Int. Conf. on Distributed Computing System, Toronto, Ontario, Canada, pp. 49–56
15. Magharei N, Rejaie R (2006) Understanding Mesh based Peer-to-Peer Streaming. In: Proc. of the 16th International Workshop on Network and Operating Systems Support for Digital Audio and Video, Newport, RI, USA
16. Tran DA, Hua K, Do T. ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. In: IEEE INFOCOM, Mar. 2003, pp. 1283–1292 25
17. Magnetto A, Gaeta R, Grangetto M, Sereno M (2010) TURINstream: A Totally pUsh, Robust, and effIcieNt P2P Video Streaming Architecture. In: IEEE Transactions on Multimedia, vol. 12, no. 8, pp. 901–914
18. Li J, Chou PA, Zhang C (2005) Mutualcast: An Efficient Mechanism for One-To-Many Content Distribution. In: Proc. of the ACM SIGCOMM ASIA Workshop, Beijing, China
19. Celebi ME, Kingravi HA, Vela PA (2013) A Comparative Study of Efficient Initialization Methods for the K-Means Clustering Algorithm. In: Expert Systems with Applications, 40(1): 200210
20. Hamerly G, Elkan C (2013) Learning the K in K-means: in: proc. Of the 7th annual conference on neural information processing systems
21. Zhu S, Wang D, Li T (2010) Data clustering with size constraints. In: Knowledge-Based Systems 23(8), 883889
22. Montresor A, Jelasity M (2009) Peersim: A scalable P2P simulator. In: Proc. of the Int. Conf. on Peerto-Peer Computing, pp. 99–100, Seattle, WA
23. Bonani M, Furlan DA (2010) Kademlia Module for PeerSim. In: Technical Report, University of Trento
24. López-Fuentes FA (2009) Video multicast in peer-to-peer networks. 138 pp, Verlag Dr. Hut, Munich
25. Peterson L, Roscoe T (2006) The Design Principles of PlanetLab. In: Operating Systems Review (OSR), Volume 40, Num. 1, 1116

Rogelio Hasimoto-Beltran is a researcher in Computer Sciences at the Center for Research in Mathematics-CIMAT in Guanajuato, México. He received his B.S. in Oceanology (with honors) from the University of Baja California, Mexico, in 1985 and his M.S. in Computer Science from the Center for Scientific Research and Higher Education at Ensenada (CISESE), Mexico in 1990. He received the Ph.D. degree in Computer and Electrical Engineering from the University of Delaware, USA in 2001. After his Ph.D., he spent two years at Akamai Technologies (a leader enterprise in Multimedia Content delivery) as a Senior Software Engineer. In February of 2003, Dr. Hasimoto joined the Department of Computer Sciences where he was tenure associate researcher and promoted to Full-Time researcher position. He has been visiting associate professor at the University of Illinois at Chicago (UIC) during 2009–2010. Dr. Hasimoto has published over 40 technical papers in refereed conferences and journals in the area of image processing, computer vision, and multimedia networks. His current research interest includes Robust Multimedia Communication, Error Concealment, Face Detection and Recognition, and Chaotic Encryption. E-mail: hasimoto@cimat.mx. Address: Jalisco S/N, Col. Valenciana, Guanajuato, Gto, México. 36,023

Francisco de Asís Lopez-Fuentes received the B.Sc. degree in electrical engineering from Oaxaca Institute of Technology, Oaxaca, México, in 1988. He received the M.Sc. degree in Computer Science minoring in Networking from Monterrey Institute of Technology (ITESM), Atizapán, México, in 1998. From 2003 to 2008 he was a member of the research staff of the Media Technology Group in the Institute for Communication Networks at the Technische Universität München (TUM), Munich, Germany, where he received the Engineering Doctorate in 2009. In December 2008, Dr. Lopez-Fuentes joined the Department of Information Technology of Universidad Autónoma Metropolitana-Cuajimalpa, México City, México, where is currently an associated professor-researcher for Networks and Distributed Systems. From 1996 to 2003, he was a professor-researcher in the Institute of Electronic and Computer Science at the Technological University of Mixteca, Huajuapan, México. From 1988 to 1994, he was a design engineer in the Engineering and Software Development Department at Siemens, México. His current research interests are in the area of networked and interactive multimedia systems, peer-to-peer networks, distributed systems as well as network security. E-mail: flopez@correo.cua.uam.mx. Address: Av. Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa, Delegación Cuajimalpa de Morelos, Ciudad de México, C.P. 05348.

Misael Antonio Vera-Lopez received the B.Sc. degree in information systems and technology from Universidad Autónoma Metropolitana-Cuajimalpa in 2015. During his B.Sc. studies he worked on simulation for peer-to-peer networks and video distribution.

Dynamic Network Coding for Collaborative Multi-source System

Francisco de Asís López-Fuentes and Javier Mendoza-Almanza
Department of Information Technology
Universidad Autónoma Metropolitana Cuajimalpa (UAM-C)
México City, México
flopez@correo.cua.uam.mx, javier-lvr09@hotmail.com

***Abstract*— Network coding is a promising technique in the field of Information Theory used to improve performance of communication networks. Several benefits related to energy savings or increase throughput have been reported in different areas when network coding is used. This paper presents a dynamic network coding approach for a collaborative multi-source system. Peer-to-peer paradigm is used to build our collaborative network between nodes in a dynamic way. Peers are synchronized by a coordinator server, which is responsible for assigning dynamic roles to the nodes that are inside the system during the network coding process. Coordinator server also must ensure that the network coding process is completed. Likewise, multiple sources are created to synchronize the nodes in terms of the contents shared by them.**

Keywords*— *network coding, peer-to-peer networks, collaboration systems, multi-source, distributed systems, information theory

I. INTRODUCTION

Currently, multimedia content are generated from different means such as social networks, mobile devices, etc. Nevertheless, multimedia contents consumes a large amount of resources in the system infrastructure. Under this scenario collaboration between nodes in order to build more robust network infrastructures is required. However, most of current infrastructures are centralized and collaboration between nodes is limited. Peer-to-peer (P2P) paradigm has emerged as a promising solution to improve the collaboration between nodes. A P2P network is deployed over a physical network as an overlay network. We are not need to change the physical connections in our network because the links in a P2P networks are logical connections established via TCP or HTTP. On the other hand, traditional collaboration between nodes has been mainly for sharing files, but the current multimedia contents often are large (e.g. movies) and these require large storage spaces. To deal with this problem, content can be fragmented into several pieces and these pieces can be located in different nodes or peers. Thus, large multimedia contents can be recovered from multiple sources. In such a way that multi-source schemes help to alleviate the unpredictability congestion in the Internet, because different parts of a big digital content can be recovered from different sources and communication links instead of a single source and one communication links. Also the problem of the single point of failure can be avoided. Multi-source schemes are taking relevance and several multi-source models have been proposed as an alternative to provide smooth video delivery [1], [2]. This paper introduces a dynamic network coding concept for video transmission from multiple sources to multiple sink under a cooperative environment. Our work exploits the benefits introduced by the P2P networks, such as load balancing and distribution of duties between all participating peers. We also exploit that a peer has characteristic to be a server and a client at same time in order to try to implement a balanced network coding between all participating peer. Thus, cooperation between nodes is not limited to their storage capacities, but that this cooperation is extended to their processing and uploading capacities. To offer video quality a system typically requires to have a high throughput and low latency in order to transmit video with high data rate. Network coding can help to reach this objective. In a network using network coding, the intermediate nodes encode the received packets from the source and forward these encoded packets to the end nodes [7, 8, 9]. However, the assignment of the nodes that perform the coding is static. In other works, each node in the network coding scheme always has a unique and static role during all network coding processes. In this work, we propose a dynamic coding scheme; in such a way that network coding can be done by all the participating nodes in the content distribution system. To evaluate our dynamic network coding approach we have implemented a prototype in Linux where the communications between all nodes are established via TCP (Transmission Control Protocol). Results obtained from our dynamic network coding show best benefits respect to results obtained from traditional network coding. We evaluated both approaches using network coding based on XOR operation.

The rest of this paper is organized as follows. Section II introduces basic concepts about network coding. Then, we describe some research works about network coding in Section III. In Section IV we give a description about our network coding model, and its implementation over a P2P network. We describe static and dynamic approaches. The performance of our model is evaluated in Section V. Conclusions and future work are given in Section VI.

II. NETWORK CODING CONCEPT

Network coding is a technique proposed by Ahlswede at al [3] to improve the information rate of a network communication. Network coding does coding at the intermediate nodes in order to increase the flow of packets considering the limit capacity of the links. Let's to consider the butterfly network shown in figure 1 to explain basic concept of network coding. There are a source node and two receiver nodes. In figure 1a we can see that the capacity of each edge is 1, which means that each links can transmit one data unit per unit time only. Figure 1b shows a source S sending two bits b1 and b2 to receiver nodes R1 and R2 simultaneously. Both receivers R1 and R2 must receive both bits b1 and b2. Node 1 broadcast bit b1 to node 3 and receiver R1, while node 2 broadcast bit b2 to node 3 and receiver R2. We can see that intermediate node 3 only receive and forward the bits received from nodes 1 and 2. In this case, link between node 3 and node 4 needs two time units to transmit bit b1 and b2 to node 4. Finally, figure 1c shows implementation of network coding in intermediate node 3. Here, network coding is denoted by the operator $\oplus$ (XOR). Link between node 1 and receiver R1 transmits bit b1, while link between node 2 and receiver R2 transmits bit b2. In this case, receive R1 can recover bits b1 and b2, but b2 must be retrieved from $b1 \oplus b2$ using bit b1 previously received from node 1. Also, R2 can recover b1 and b2, but in this case b1 must be retrieved from $b1 \oplus b2$ using bit b2 previously received from node 2. We can see how multicast rate is increased from a 1 bit/time unit to 2 bits/time unit by using network coding.

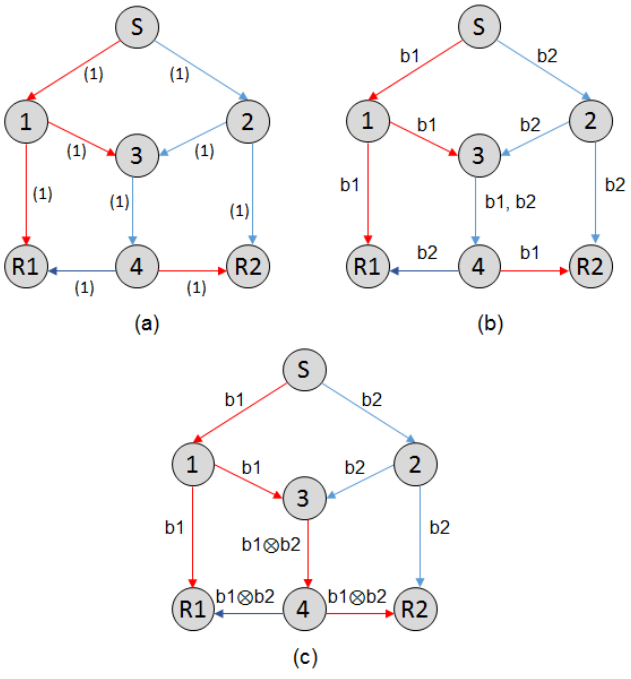


Figure 1. Example of a communication network. a) Capacity of the edges, b) Traditional approach and c) approach with network coding

III. RELATED WORK

Several applications related with communication networks using network coding are reported in the literatura. Network coding have been implemented in networks, wireless networks [18, 19], MANETs, ad-hoc networks, sensor networks [6], wireless mesh networks [10], and quantum networks [20, 21]. In most of these cases, network coding provides different benefits in the communication networks such as reliable broadcasting, efficient data dissemination, saving bandwidth, improved system throughput, reduced delays and and recovery data. Different research works about network coding are reported in the communication network literature. Some active research areas of network coding are sensor networks [11], security issues [17], video streaming and content distribution [8]. A scheme for large scale content distribution using network is presented by Gkantsidis et al in [8]. Authors report that network coding helps to increase the throughput in a system between 20% and 30% in comparison with a system where coding is done in the source. Network coding provides robustness to a system in extreme situations such as departure of nodes or sudden server. Also, systems using network coding have a better performance than using unencoded blocks or erasure codes. Sundararajan et al. [9] focus network coding techniques on wireless networks. In this case, authors evaluate the impact of network coding on the TCP/IP (Transmission Control Protocol/Internet Protocol). To this end, they introduce a new layer between the transport layer (TCP) and network layer (IP), but without modifying the congestion control mechanism. Their results show that TCP with network coding allows that a wireless network slight increases its throughput and have greater robustness in a packet loss situation. A work about network coding focused toward sensor networks is SenseCode [11]. This work address network coding to solve problem related to reliability and transmission energy in a sensor networks. SenseCode is evaluated in term of its reliability and transmission energy and the obtained measurements are compared with the results obtained using CTP (Collection Tree Protocol) [12] on the TOSSIM platform [13]. Results show that using network coding sensor network can reach a balance between energy efficiency and reliability. Security also is a field where network coding has been a positive impact. For example, Lima et al. [14] use network coding to design a secure architecture for video transmission over wireless network in scenarios with losses. Network simulator NS-2 is used to evaluate this architecture. Authors reports several benefits by using network coding in packet encryption such as smaller packets than those obtained by the traditional method, and less packet loss. Where less packet loss means to have a better video quality. A work about network coding addressed to video streaming system is presented by Nguyen et al. in [16]. In this case, network coding is studied in video broadcasting application for wireless network, and the author are interested in developing an optimal scheme for retransmission of lost packet using erasure codes with network coding. Other authors have found that using network coding in content distribution systems it is possible to achieve optimal multicast rate [3]. Inspired in these previously works, in this paper, we try to extend the network coding concept based on XOR operation from a static approach to a dynamic approach for multi-source systems.

IV. PROPOSED MODEL

Our model uses different nodes, which are peers and servers. Servers store information about the peers and files shared by them. Due to this fact the servers are also called source nodes. Our proposed model with multiple sources is shown in figure 2. Each source distributes its initial content to different requesting peers, and sources are independent of each other. Each source distributes a type of file. Thus, video, music, photos and PDF files are distributed from sources S1, S2, S3 and S4, respectively. Network coding is implemented in this multi-source scheme in order to improve the distribution time. Each server must respond all received requests. Inside the server there is a matrix which stores the IP address of all nodes and the files shared by them. When a server receives a request, it creates a thread to responds this request. For each request a thread is created to respond all requests at the same time (in a concurrent way). This thread establishes a communication with the peer that wants to share files or synchronize its communication with the server.

In our architecture, a peer establishes communication with different sources and different peers in order to share content and to have information about the contents that are shared within the network. Each peer has different matrices with information about the peers that are connected and the files shared by them. For each source the peer has a different matrix. In these matrices the information is organized using the IP address of the peer and the names of the contents to be shared. When a peer wants a new content from another peer these matrices must be synchronized with the different sources. To request a file, a peer creates different threads to synchronize their matrices and to update information only of the peers that are connected and the files shared by them. Once the matrices are updated, the peer establishes communication with peer where wished file is available, and create a thread which establishes communication with that peer to receive the wished files. A peer can create different threads to request a content to any the different peers within the network and likewise can distribute content to all peers that request a file from it. In [4] is presented a detailed explanation about the multi-source scheme used in our experiments.

A. *Static network coding*

Our multi-source architecture is enabled to run network coding. To do that, a peer has to involve other peers within the network and use different threads with specific tasks. Thus, peers can be organized as is shown in figure 3. In this scheme, we have used a traditional network coding approach, which in this work we have called static network coding. This means that each peer in the architecture has only a specific task to do during all network coding process. Figure 3 shows this scenario. Here, peer P1 receives video1 from S1 and video2 from S2. After this, peer P1 creates an encoded file based on the two received files. To this end, peer reads bit by bit from both files and applies the XOR operation for each bit that reads from both files, and the result is saved in the encoded file. Sink peers P3 and P4 receive the encoded file and create a thread to one of the source peers to request the original file. P3 requests the video2 from peer S2 and P4 requests video1 from S1. Therefore, one peer will have the encoded file and one of the

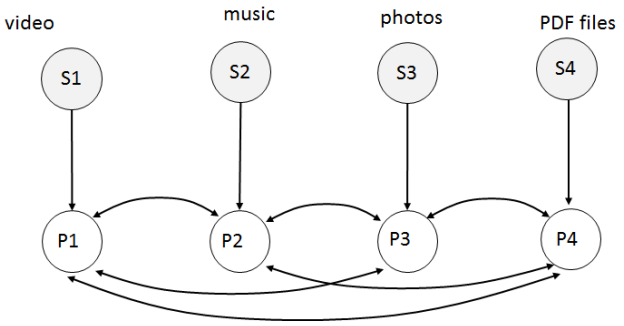


Figure 2. Example of a multi-source scheme

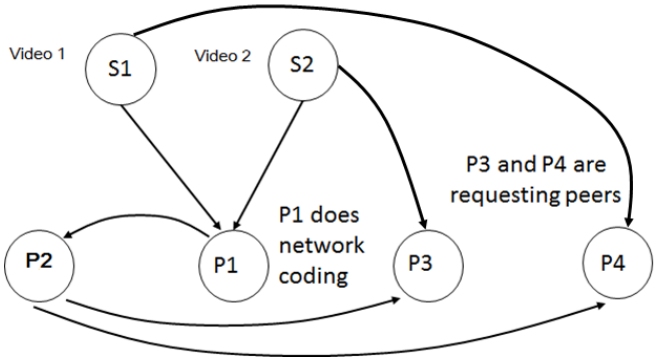


Figure 3. A multi-source scheme organized to implement network coding

original files while the other peer will have the encoded file and another original file. After this, peers P3 and P4 open both files bit by bit and create a new file which will contain the original file. For each bit that is read from both files, the peers apply the XOR operation and the result is stored in the created file. A detailed explanation about a practical static network coding implementation based on XOR operation is given in [5].

B. *Dynamic network coding*

Unlike the static network coding, in dynamic network coding model we have incorporate a coordinator server, which assigns the different roles to the peers in order to avoid its saturations and reach a dynamic collaboration in the system. Figure 4 shows a collaborative scheme with a coordinator server. In this case, coordinator server there are two matrices. First matrix stores the address of all the different peers within the network and second matrix stores the role of peers during the network coding process. In our scheme, each peer has six different roles which are source1, source2, cipher, distributor, receiver1 and receiver2. Second matrix also saves role history for each peer in order to avoid its saturation. Figure 5 shows these six different types of roles in each peer.

Dynamic network coding works as follows. When the coordinator server receives a request, it automatically assigns

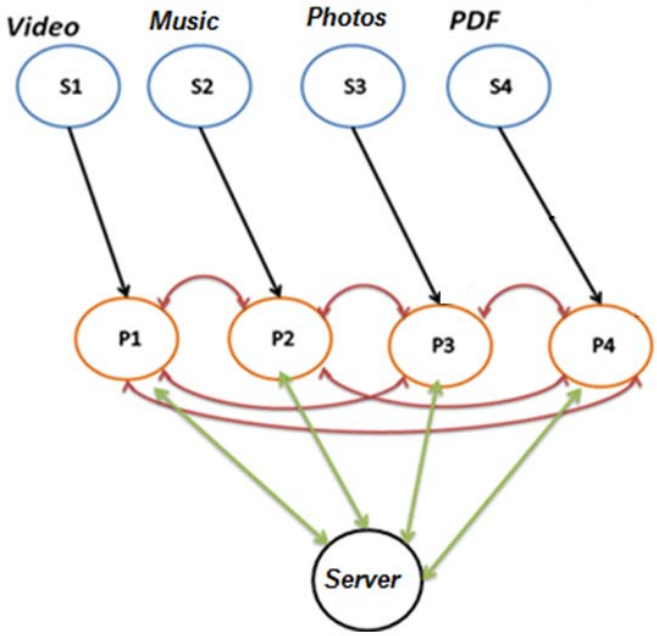


Figure 4. A multi-source scheme with a coordinator server

roles to the peers that are within the network. For this, first we need to get the IP addresses of the receivers. This is possible when a peer receives a request from the first receiver and the peer that makes the request sends the IP address of the other receiving peer with information of the sources where the wished file by the receiving peers is available. Once the sources and receivers are available, the server obtains the peers that are not present in the running processes and makes two selections at random to obtain two IPs. Then, one peer is assigned as encoder while the other peer is assigned as a distributor. When all peers have been obtained, the server stores information from them in the process matrix and creates threads for the source peers to notify their role. When peers finish their roles within the system, the server creates a thread for the encoder peer which sends a notification to the server who creates a thread for the distributor peer. Then, server creates two threads for the receivers peers in order to notify their role and verify that they can obtain both original files. Finally, the server frees all peers that were inside the process.

V. Evaluation and Results

To evaluate our dynamic network coding for a collaborative multi-source system, we have realized different tests which consisted in executing two network coding processes in a network formed by six peers and four files. All files have the same size of 32 MB. We have compared the results obtained from our dynamic network coding approach with results obtained using the static network coding approach.

Our first experiment evaluates a multi-source collaborative system using static network coding. In this test the roles of the peers are indicated in Table I.

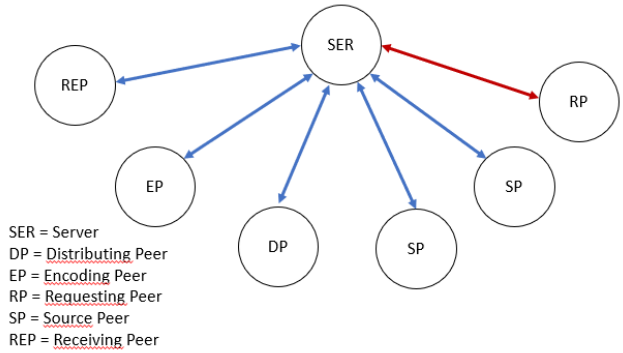


Figure 5. Interaction between peers and the server for dynamic network coding

TABLE I. Role of the peers using static network coding

Peer	*Role (process 1)*	*Roles (process 2)*
Node 1	Source 1	Source 2
Node 2	Receiver 1	Receiver 2
Node 3	Distributor	Distributor
Node 4	Encoder	Encoder
Node 5	Source 2	Source 1
Node 6	Receiver 2	Receiver 1

In table I we can see that some peers have the same role, especially the peers with the roles of distributor and encoder. The responsible peer for playing the role of coding can present saturation by having the same role assigned to it at the same time, which may imply that in case there are more network coding processes at the same time this node could stop working. The required time for each process is resumed in table II.

TABLE II. Required time for each process

Process	*Time (seconds)*
1	10.2
2	11.1

Our second experiment evaluates a multi-source collaborative system using dynamic network coding. In this scenario, network coding is performed in a dynamic way by the coordinating server. In the table III, we can see the roles that were assigned to the peers. Although some peers play robust roles in both processes, they do not do it at the same time, which implies that they do not become saturated, as well as the roles of encoder and distributor are dynamically assigned. Results obtained in this evaluation are shown in table IV.

Comparing both systems, we can get benefits by incorporating a coordinator server in the dynamic network

coding approach. We can see that process 1 reduces its time in around 9%, while process 2 reduces its time around 25%. It is important to mention that the sources and the receivers are roles that do not depend on the coordinator server. In other words, the requesting peer defines these roles implicitly when requesting the files it wants and when mentioning the second receiving peer. However, the roles of encoder and distributor are completely dynamic, which implies an improvement within the system. Also the coordinator server when managing the roles of the peers implies a lot of gains in different areas. It avoids saturating the peers which implies that it prevents them from working and allows dynamism within the peers improving times.

TABLE III. ROLE OF THE PEERS USING DYNAMIC NETWORK CODING

Peer	*Role (process 1)*	*Role (process 2)*
Node 1	Source 2	Source 2
Node 2	Receiver 1	Receiver 2
Node 3	Distributor	Encoder
Node 4	Source 1	Source 1
Node 5	Encoder	Distributor
Node 6	Receiver 2	Receiver 1

TABLE IV. REQUIRED TIME FOR EACH PROCESS

Process	*Time (seconds)*
1	9.3
2	8.6

VI. CONCLUSIONES

Network coding has had a positive impact on modern communication networks, and several research areas have reported different benefits when using network coding in their respective projects. In this work, we have presented and evaluated a network coding scheme base on XOR operation with a dynamic assignation of roles. This mean all nodes in the networks can perform network coding. This concept introduces several benefits in a collaborative system because the nodes not only share files but also processing capacity is shared by all participating peers in the system. There is also a better load balancing respect to processing capacity of all peers. Results report benefits respect to reduction of distribution times of contents in a collaborative system. Our work is in progress. As future work, we plan to combine machine learning techniques with our dynamic network coding approach in order to improve the system performance by doing an optimal tasks assignment in each peer. We also plan to implement algorithms to synchronize the matrices and pass all the matrices to a distributed database. The system can also be made more robust by implementing some security mechanisms.

REFERENCES

[1] H. Pucha, G. d. Andersen, and M. Kaminsky, “Exploiting Similarity for Multi-Source Downloads Using File Handprints,” 4th USENIX NSDI '07, Cambridge, MA, USA April 2007.

[2] F. A. López-Fuentes, and E. Steinbach, “Multi-source video multicast in peer-to-peer networks,” IEEE International Symposium on Parallel and Distributed Processing (IPDP 2008), pp. 1- 8, Miami, FL, USA, 2008.

[3] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, “Network information flow,” IEEE Trans. Inform. Theory, 46, 1204–1216, 2000.

[4] J. Mendoza-Almanza, and F. A. López-Fuentes, “Collaborative Multi-source Scheme for Multimedia Content Distribution,” WITCOM 2016, Silao, Gto. México, 2016.

[5] J. Mendoza-Almanza, F. A. López-Fuentes, R. Hasimoto-Beltran, “Practical Network Coding for Multi-source Scenarios,” Smart Technology, Springer, pp. 141-148, 2018.

[6] L. Keller, E. Atsan, K. Argyraki and C. Fragouli, “SenseCode: Network Coding for Reliable Sensor Networks,” Tech. Rep. 2009, Ecole Polytechnique Federale Lausanne (EPFL). Last updated July, 2010.

[7] P. Chou, Y. Wu and K. Jain, “Practical network coding,” 51st Allerton Conf. Communication, Control and Computation, Monticello, IL, USA, 2003.

[8] C. Gkantsidis and P. R. Rodriguez, “Network Coding for Large Scale Content Distribution,” IEEE INFOCOM 2005, Miami, FL, USA, pp. 2235–2245, 2005.

[9] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher and J. Barros, “Network coding meets TCP,” IEEE INFOCOMM 2009, Rio de Janeiro, Brazil, 280–288, 2009.

[10] F. Jamil, A. Javaid, T. Umer, E. and M. H. Rehmani, “A comprehensive survey of network coding in vehicular ad-hoc networks,” Wireless Networks, 23(8), pp. 2395–2414, 2017.

[11] L. Keller, E. Atsan, K. Argyraki, and C. Fragouli, “SenseCode: Network Coding for Reliable Sensor Networks”, Tech. Rep. 2009, Ecole Polytechnique Federale Lausanne (EPFL). Last updated July, 2010.

[12] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss and P. Levis, “Collection tree protocol”, 7th ACM Conf. Embedded Netw. Sensor Sys. SenSys, Berkeley, CA, USA, 2009.

[13] P. Levis, N. Lee, M. Welsh, and D. Culler, “TOSSIM: Accurate and scalable simulation of entire TinyOS applications”, First ACM Conf. Embedded Netw. Sensor Sys. SenSys, Los Angeles, CA, USA, 2003.

[14] L. Lima, S. Gheorghiu, J. Barros, M. Medard, and A. Toledo-López, “Secure network coding for multi-resolution wireless video streaming”, IEEE J. Selec. Areas Commun., vol. 28, no. 3, pp. 377–388, 2010.

[15] S. McCanne, S. Floyd, and K. Fall, “The network simulator, ns-2” [Online]. Available: http://www.isi.edu/nsnam/ns

[16] D. Nguyen, T. Nguyen, and X. Yang, “Multimedia wireless transmission with network coding,” Packet Video, Lausanne, Switzerland, pp. 326–33, 2007.

[17] F. Chen, T. Xiang, Y. Yang, and S. S. M. Chow, “Secure cloud storage meets with secure network coding,” IEEE INFOCOM, 2014.

[18] D. P. Wu et al., "Dynamic coding control in social intermittent connectivity wireless networks", IEEE Trans. Veh. Technol., vol. 65, no. 9, pp. 7634-7646, Sep. 2016.

[19] C. Xu, P. Wang, C. Xiong, X. Wei, G.-M. Muntean, “Pipeline network coding-based multipath data transfer in heterogeneous wireless networks”, IEEE Trans. Broadcast., vol. 63, no. 2, pp. 376-390, Jun. 2017.

[20] M. Epping, H. Kampermann and D. Bruß, “Robust entanglement distribution via quantum network coding”, New Journal of Physics, DPG/IOP, Vol.18, pp. 2016.

[21] H. Kobayashi, F. L. Gall, H. Nishimura, and M. Rötteler, “Perfect quantum network communication protocol based on classical network coding,” IEEE International Symposium on Information Theory, pp. 2686–2690, 2010.

Practical Network Coding for Multi-source Scenarios

Javier Mendoza-Almanza[1], Francisco de Asís López-Fuentes[1(✉)], and Rogelio Hasimoto-Beltran[2]

[1] Departament of Information Technology, Universidad Autónoma Metropolitana-Cuajimalpa (UAM-C), Av. Vasco de Quiroga 4871, Cuajimalpa, Santa Fe, 05348 Mexico City, Mexico
flopez@correo.cua.uam.mx
[2] Centro de Investigación en Matemáticas (CIMAT) A. C., Jalisco S/N, Col. Valenciana, 36023 Guanajuato, Gto, Mexico
hasimoto@cimat.mx

Abstract. Network coding is a proposed technique for improving network capacity. This novel concept has been mainly oriented to increase throughput and reliability of communication networks. Network coding is implemented in the intermediate nodes before forwarding the encoded packets to the end nodes. The received packets are decoded in the end nodes in order to recovery the original transmitted data (video data in our case). In this work, we investigate the performance of network coding in collaborative multi-source scenarios with heterogeneous resources (video, image, audio, pdf files). Collaborative multi-source schemes are very important for critical multimedia services because multimedia content consumes an important amount of resources in the communication networks. A multi-source scheme is a useful solution when different parts of a multimedia content is generated or stored in two or more sites. Our evaluation compares the performance of a P2P networking with network coding against a client-server communications. Results show great benefits on using network coding scheme in collaborative multi-source scheme.

Keywords: P2P networks · Content distribution · Distributed systems

1 Introduction

Several content distribution infrastructures have emerged in response to high demand for multimedia contents. Most of these infrastructures have based on central servers, which exhibit several limitations and a reduced collaboration between requesting nodes. Because the multimedia content consumes a large amount of resources in a communication system, collaboration among requesting nodes play an important role. In this context, peer-to-peer (P2P) networks have emerged as practical solution for constructing collaborative infrastructures. P2P systems have generated great interest in the research community who find in these systems a fast and efficient way to deliver movies, music or software files [1–3]. In a P2P system, the users interact directly as a way to exchange their resources and services through the Internet. Multimedia content

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2018
F. Torres Guerrero et al. (Eds.): MTYMEX 2017, LNICST 213, pp. 141–148, 2018.
https://doi.org/10.1007/978-3-319-73323-4_14

requires large storage spaces, and large multimedia contents (e.g. movies) often exceed the storage capacity of a personal device in traditional centralized network architecture. A P2P system on the other hand, distributes its load and duties between all participating peers. Multi-source schemes are used in order to solve these requirements. Multi-source schemes are also required when content is produced from multiples sites [4, 5]. For example, in soccer match. On the other hand, in a multi-source scheme, each source can distribute different video sequences to all requesting peers. How much the source can redistribute depends on the available upload capacity. At the same time, each requesting peer forwards the video directly received from a source to the rest of the peers. Again, the amount of redistributed content depends on the peers' upload capacity. The upload capacities of the sources are divided equally among the different video streams. This paper presents a practical implementation of network coding using XOR operations in a multi-source scenario based on P2P infrastructures [4]. Multi-source scheme disseminates multimedia contents from multiple sources to multiple requesting peers. Initially, sources distribute the original content to an intermediate peer, where network coding is applied. After this, the intermediate node sends the encoded content to the requesting peers. Original content is recovered in each requesting peer by using XOR operation to decode the encoded packets.

The rest of this paper has the following organization. In Sect. 2, we give a brief introduction to network coding. Section 3 presents a P2P multi-source scheme in which network coding is tested. A practical implementation of our multi-source scheme is described in Sect. 4. Our paper concludes in Sect. 6.

2 Network Coding Overview

Ahlswede et al. [6] introduced network coding as a new technique related to information flow in the communication networks. This technique employs coding at the intermediate nodes in the network to increase the flow without exceeding the links capacity. Network coding is inspired by the Max-Flow Min-Cut theorem, which states that [6]: "The maximum amount of flow from the source to the destination nodes equals the minimum capacity required to remove flow from the network that cannot pass from source to destination". Authors represent the network as a directed graph $G = (V, E)$, where the network nodes are represented by V, and the edges E represent the communication links. The link capacity is of one data unit per unit time. The source node is a node without any incoming edge. In a single-source multicast session, the source node $s \in S$ transmits information at rate R to all receivers $t \in T$, and the maximum multicast information rate in this scenario can be achieved only by allowing coding at intermediate nodes [6]. This maximum multicast rate can be given by finding the capacity through the previously described Max-Flow Min-Cut theorem, which relates the maximum information flow through a network with the minimum cut capacity.

Figure 1 presents a scenario of a butterfly network with a source node and two receiver nodes. Figure 1(a) shows the capacity of each edge. We can observe that the value of the maximum flow of S to any receiver, either $R1$ or $R2$ is equal to two. Therefore, in Fig. 1(b) source S can send two bits, $b1$ and $b2$, to $R1$ and $R2$ simultaneously. In this scheme, each intermediate node only replicates and sends out the bit(s)

received from upstream. On the other hand, Fig. 1(c) shows the same network configuration, except that network coding is implemented. Here, the operator $\oplus$ denotes the sum module 2. Thus, the receiver $R1$ can recover the two bits, $b1$ and $b2$, except that $b2$ must be retrieved from $b1 \oplus b2$. Similarly, $R2$ can recover the two bits. In this example, network coding is applied at node 3. Another important point is that the multicast rate increases, because in traditional transmission the rate is 1 bit/time unit, whereas applying network coding the rate increases to 2 bits/time unit.

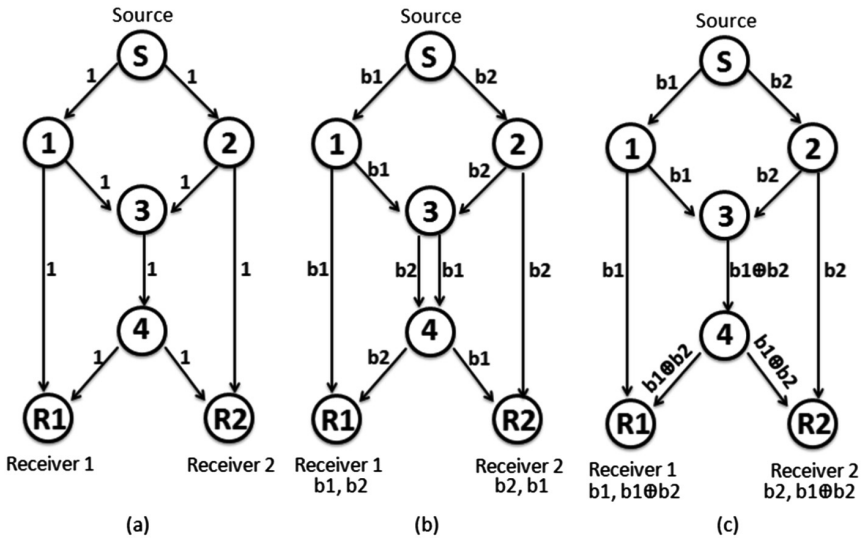


Fig. 1. Communications network: (a) capacity of the edges, (b) traditional approach and (c) approach with network coding.

In contrast to traditional ways of operating a network that try to avoid collisions of data streams as much as possible, this elegant principle implies a plethora of surprising results. One of the most exciting opportunities of the approach is the use of random mixing of data streams, thus freeing up the symmetrizing properties of random coding arguments in network analysis.

The most common benefits of using network coding in a communication network are [7, 8]: saving bandwidth, improved system throughput, and reduced delays. Multimedia application requires high data rates, low-latency and low packet loss rates, which represent a significant challenge for the design of future network architectures. In this scenario, network coding can help to improve the performance of the multimedia systems.

3 Multisource Scheme

Multi-sources model studied in this work is shown in Fig. 2. Each source S distributes its initial content to different requesting peers. Our solution assumes that sources are independent to each other. Different videos are distributed from each source. Source S1 distributes video 1, while S2 distributes video 2. Different types of files such as music files, photos files, PDF files, and other type of files can be distributed in this multi-source scheme. Network coding is implemented in this multi-source scheme network in order to improve the distribution time.

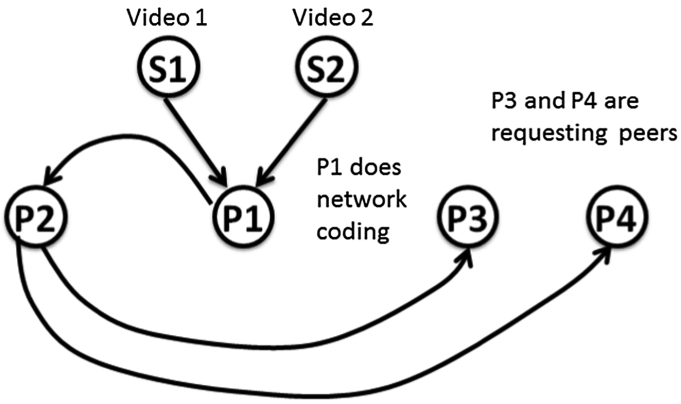


Fig. 2. Multi-source model

Initially, each server sends its content to requesting peer. After a peer receives a type of file, it can be distributed to the rest of requesting peers. Thus, each source distributes only its files in the first stage. In the second stage, files are redistributed among all requesting peers. Each server has two functions. First function is to distribute the original content, and second function is to maintain information about peers with distributed files. This information is stored in a database in each server, which is periodically updated. In this way, a server reduces its workload. When new peers arrive to system, information about IP address of each peer and its shared content can be obtained from any server.

In a multi-thread application, different processes can be executed at the same time concurrently. The number of processes is variable and depends on the amount of connections that are established. In this work, we have used multi-thread to improve the performance of our system. Coordinator peer is the main thread in each peer, and it is responsible for synchronization and control of the others threads. Client_thread_1 is responsible for connecting with the main servers. Thus, this thread requests a content and receive it from the source. Client_thread_1 also receives the table with information of IP address and name (ID) of all requesting peer in the system from the source. Each peer uses the database with IP address and name of peers to establish the network structure. A peer can receive multimedia contents from the main sources.

4 Implementation

In this section, we implement our proposed multisource network coding scheme by considering two main objects, *entities* and *servers*. The *entities* are the counterpart of peers (as discussed in Sect. 3), and the *servers* are the ones that store information about connected peers and files to be shared in the network. There are although different kind of servers depending on the information and type of file they are capable to distribute and store. There is one server for distributing video content, another for distributing images, or music, or pdf files, etc. In this way, we are representing the concept of multisource with different kind of source-data servers.

Each server manages an information matrix that stores the IP addresses of all nodes (servers and peers) in the network, along with their corresponding content or files that can be shared to the rest of the nodes. When a server receives either a synchronization or file-sharing request from a node, a communication thread is created from the server to the node for each request and for each file to be shared. Immediately after this file sharing process, the server sends to the node its information matrix, so that the node can communicate with the different servers in the network to request a particular content. If the node just wants to synchronize, the server only shares the information-matrix containing the updated information of the current content and nodes in the network. As mentioned above, in our scheme each server behaves in the same way but with different type of data or content.

A peer is an entity or application running on each node of the centralized P2P network. The peer application sends and receives files at the same time, in addition to performing the network coding process. A peer node in our architecture, establishes communication with all sources and servers to distribute and receive content, as well as to gather information about all contents to be shared in the network. The content synchronization among all peer entities is permanent in order to keep all the information matrix updated in the entire network. Once the information matrix has been updated, peers can request content again. However, in this paper our tests are made with a static P2P infrastructure, where roles for each peer are fixed.

An important function of peer nodes is network coding. In order to perform this task, several peer entities are involved in the process for which, each peer has to create different threads with concrete tasks. Assume that two data bits a and b are multicast from sources A and B to nodes E and F as depicted in Fig. 3. We assume that nodes E and F start a request, which is attended by the servers A and B, and the intermediate nodes. Nodes A and B send file bits a and b respectively to common connected peer C. In this operation, C is the one that initiates the file request by creating two independent thread process, one for each peer. C initiates the coding process of the received files received by applying the XOR operation on the two bit-streams to create new coded file represented by $a_i \oplus b_i$. When the length of the involved files is different, the file counter with the smallest length is passed on to intermediate C and D) and end peers (E and F) for the coding and decoding processes respectively. A subset of the peers who received the coded files, requests the original file from peers that received the original files from the source. In this case, E receives file a from A and F receives b from B. The coded

and original files are XOR to obtained the other original file, b in the case of peer E and a in the case to peer F. In Fig. 3, *t* indicates the smallest file size.

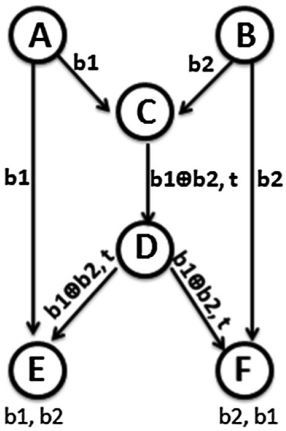


Fig. 3. Network coding application in multi-source model

If a particular peer wants to be excluded from participating on the network coding and receiving content, it only maintains a thread process that respond requests from other peers (e.g. a content).

5 Evaluation

We have evaluated the operation of our prototype in a local network in our campus. Both servers and peers work correctly. Files are sent from each dedicated server and each requesting peer correctly, and network coding is done in the intermediate nodes. In our experimental set-up, we implemented two different performance tests for content distribution: (a) client-server (where only one server serves the content to all clients) vs. P2P architectures, and (b) client-server architecture vs. P2P with network coding architectures. In the first case (a), we compare the delivery time in order to prove the content distribution load between these two architectures. Table 1 presents the time it

Table 1. Distribution time in a client-server architecture

	Client 1	Client 2
File 1	1.54	1.59
File 2	1.56	2.02
File 3	2.02	2.03
File 4	2.04	2.04
Total	7.56	8.08

takes to distribute (from sender to receiver node) 4–28 MB files in a client-server architecture:

Next, Table 2 presents the corresponding delivery times for a P2P architecture:

Table 2. Distribution time in a P2P architecture

	Peer 1	Peer 2
File 1	0.40	0.41
File 2	0.41	0.38
File 3	0.31	0.24
File 4	0.33	0.33
Total	2.25	2.16

Results show that P2P architecture delivers the content in a shorter time since each peer behaves as both client and server, while the client-server architecture may saturate the server if many requests are performed at the same time.

We now compare the P2P with network coding against client-server architecture (Table 1). Let us first show the transmission of two files in a pure P2P architecture (Table 3) against P2P with network coding (Table 4):

Table 3. Distribution time in pure P2P architecture

	Receiver 1	Receiver 2
File 1	0.41	0.42
File 2	0.42	0.42
Total	1.23	1.24

Table 4. Distribution time in pure P2P architecture with network coding

	Receiver 1	Receiver 2
File 1 y File 2	0.47	0.45
Total	0.47	0.45

It can be observed that P2P with network coding is the most efficient architecture for content delivery, since we are sending two files at the same time.

6 Conclusions

There is currently a high demand for multimedia content, and collaboration among requesting nodes play an important role. In this paper, we have evaluated the performance of a multi-source scheme using network coding to distribute multimedia content from many sources to many requesting peers. Implemented is deployed on a peer-to-peer network. Using this proposed approach, the sources can distribute their

workload between all requesting peers, and the system can improve its performance. Our scheme uses threads to establish collaboration connections with other peers. The number of threads is variables and depends on the amount of established connections. In each peer, a coordinator thread deals with the incoming requests and creates the rest of threads that handle the requests.

Our purpose in this work is to develop a P2P multisource network with network coding capabilities, which at this point it works fine. There exist some details to be improved, such as the use of Linear Network Coding (LNC). LNC would improve the efficiency of file transmission and would eliminate the use of some transmission channels in which we could only send ciphered buffers based on Galois fields to get back the original file. Finally, we can have a special server in the network to decide the peer nodes who will take part of the network coding task and at the same time the peer nodes who will perform the coding process at the same time.

References

1. Milojicic, D., Halogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard B., Rolling, S., Xu, Z.: Peer-to-Peer Computing. HP Labs Technical report HPL-2002-57 (2002)
2. Sayit, M., Demirci, S., Kaymak, Y., Tunali, E.: Adaptive, incentive and scalable dynamic tree overlay for P2P live video streaming. Peer-to-Peer Netw. Appl. **9**, 1074–1088 (2016)
3. Kuo, J.L., Shih, C.H., Ho, C.Y., Chen, Y.C.: Advanced bootstrap and adjusted bandwidth for content distribution in peer-to-peer live streaming. Peer-to-Peer Netw. Appl. **8**(3), 414–431 (2015)
4. Mendoza-Almanza, J., López-Fuentes, F.A.: Collaborative multi-source scheme for multimedia content distribution. In: WITCOM 2016, Silao, Gto. México (2016)
5. López-Fuentes, F.A., Steinbach, E.: Multi-source video multicast in peer-to-peer networks. In: Proceedings of the IEEE International Symposium on Parallel and Distributed Processing, Miami, FL, USA, pp. 1–8 (2008)
6. Ahlswede, R., Cai, N., Li, S.-Y., Yeung, R.W.: Network information flow. IEEE Trans. Inform. Theory **46**, 1204–1216 (2000)
7. Gkantsidis, C., Rodriguez, P.R.: Network coding for large scale content distribution. In: IEEE INFOCOM 2005, Miami, FL, USA, pp. 2235–2245 (2005)
8. Keller, L., Atsan, E., Argyraki, K., Fragouli, C.: SenseCode: Network Coding for Reliable Sensor Networks, Technical report 2009, Ecole Polytechnique Federale Lausanne (EPFL), July 2010

ISSN 1870-4069

Collaborative Multi-Source Scheme for Multimedia Content Distribution

Javier Mendoza Almanza, Francisco de Asís López-Fuentes

Universidad Autónoma Metropolitana-Cuajimalpa,
Departament of Information Technology, Mexico City,
Mexico

flopez@correo.cua.uam.mx

Abstract. Demand for multimedia contents has increased in recent years, and several distribution services have emerged. Many of these multimedia distribution services are based on central servers, which introduce several limitations related with costs, dependence, performance or scalability. This paper presents a collaborative scheme for multimedia content distribution. Collaborative infrastructures for multimedia services are critical because multimedia contents have an import consume of resources in the communication networks. P2P networks have emerged as promising solution to implement collaborative infrastructures. Multi-source schemes are a practical solution when different parts of multimedia content is generated or stored in two or more sites. We have used a P2P network to implement a practical distribution prototype of our collaborative multi-source scheme. Our evaluation shows as peers share storage capacity, contents and bandwidth capacity, while server is released from this workload.

Keywords: P2P networks, content distribution, distributed systems.

1 Introduction

During recent years, several content distribution infrastructures have emerged in response to high demand for multimedia contents. Most of these infrastructures have based on central servers, which present several limitations and present a reduced collaboration between requesting nodes. Because the multimedia content consumes a large amount of resources in a communication system, collaboration between requesting nodes play an important role. In this context, peer-to-peer (P2P) networks have emerged as practical solution for constructing collaborative infrastructures. P2P systems have generated great interest in the research community who find in these systems a fast and efficient way to deliver movies, music or software files [1, 14, 15]. In a P2P system, the users interact directly as a way to exchange their resources and services through the Internet. Multimedia content requires large storage spaces, and large multimedia contents (e.g. movies) often exceed the storage capacity of a personal device. Multi-source schemes are used in order to solve these requirements. Multi-source schemes are also required when content is generated from multiples sites. For

example, in soccer match. This paper reviews different content distribution schemes, and introduces a collaborative distribution scheme based on P2P networks. In a P2P network, a node can take the role of both a server and of a client at the same time. Thus, when a new peer arrives to the system, the demand in the system is increased, but the system's overall capacity is increased too. A P2P system distributes its load and duties between all participating peers, which is not possible in a system based on central servers.

P2P networks are becoming more and more popular today (they already generate most of the traffic in the Internet). For instance, P2P systems are very used for file sharing and distribution; some examples are Bittorrent [2], Tribbler [3], eMule [4], GridCast [11], etc. Main technical problem is that peers connect and disconnect with high frequencies, in an autonomous and completely asynchronous way. This means that the resources of the network as a whole are also highly variable, and thus, that the network must be robust face to these fluctuations. In order to face the high dynamicity of such a system, we explore a multi-path approach where (i) the stream is decomposed in some way into several flows; (ii) each client receives those flows following different paths and sent from different other clients.

On the other hand, in a multi-source scheme, each source can distribute different video sequences to all requesting peers. How much the source can redistribute depends on the available upload capacity. At the same time, each requesting peer forwards the video directly received from a source to the rest of the peers. Again, the amount of redistributed content depends on the peers upload capacity. The upload capacities of the sources are divided equally among the different video streams. In this paper, we present a collaborative multisource scheme based on P2P networks. In this paper is proposed a multi-source scheme to disseminate multimedia content from multiple source to multiple requesting peers. Initially, sources distribute the original content to each requesting peer. After the requesting peers receive the content, they can redistribute this content to other peers in a collaborative way.

The rest of this paper has the following organization. In Section 2, we present information about different multi-source schemes based on P2P networks. Section 3 presents our proposed model. A practical implementation of our multi-source scheme is described in Section 4. Our paper concludes in Section 5.

2 Related Work

Dissemination information to a large group of nodes from many sources is fundamental in many systems and applications. Multi-source P2P multicast applications recently have been used for collaborative environments such as conferencing or multi-player games. A P2P network is an overlay network formed by a group of nodes. P2P systems maintain their independence of the underlying physical network by using this overlay topology. In a P2P network, a company can disseminate information. Thus, content can be distributed to an audience without the need for any special support from the network (Jannotti et al 2000), and where the upload capacity of the participating peers is only considered to forward the content. We can create a collective organizational knowledge within the organization, or share data and application files between computers without a dedicated server. However, P2P overlays known as unstructured

and structured overlay show limitations for multi-source multicast such as scalability [6], large overhead [7] or complex protocols [8].

Several approaches for content distribution from multiple sources to a single receiver can be found in the literature. Authors in [9] exploit the similar source concept to significantly improve the download time of a file from multiple sources to one receiver. Push is proposed in [10] as an efficient generic push-pull dissemination protocol. Pulp exploits the efficiency of push and pull approaches, such that it presents achieve reasonable latency and presents a low overhead by limiting redundant messages. In a multi-source environment, sources can provide conflicting values (false or true information). To deal with this problem, authors in [12] propose Datafusion, a novel solution to find true values from conflicting information when in the system there are a large number of information sources. In [13] is proposed a framework for video delivery from multiple sources to multiple receivers using P2P networks. In this work, authors consider that sources are requesting peers too. This work introduces a concept called helper peer, which is not interested in receiving the content and just contribute their resources during distribution.

3 Proposed Model

Our proposed model with multiple sources is shown in Figure 1. Each source distributes its initial content to different requesting peers. Our solution assumes that sources are independent of each other. Different type of files are distributed from each source. Source S1 distributes video files, while S2 distributes music files, S3 distributes photos files and S4 distributes PDF files. S5 is used to distribute other type of files.

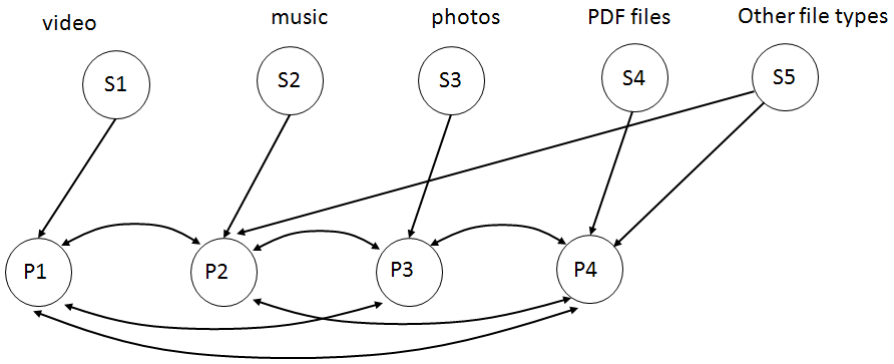


Fig. 1. Proposed model.

Initially, each server sends its content to requesting peer. After a peer receives a type of file, it can be distributed to rest of requesting peers. Thus, each source distributes only its files in the first stage. In the second stage, files are redistributed among all requesting peers. Each server has two functions. First function is to distribute the original content, and second function is to maintain information about peers with distributed files. This information is stored in a database in each server, which is periodically updated. In this way, a server reduces its workload. When new peers arrive

to system, information about IP address of each peer and its shared content can be obtained from any server.

In a multi-thread application, different processes can be executed at the same time concurrently. The number of processes is variable and depends on the amount of connections that are established. In this work, we have used multi-thread to improve the performance of our system. Figure 2 shows threads active in each peer. Coordinator peer is the main thread in each peer, and it is responsible for synchronization and control of the others threads. Client_thread_1 is responsible for connecting with the main servers. Thus, this thread requests a content and receive it from the source. Client_thread_1 also receives from the source the table with information of IP address and name (ID) of all requesting peer in the system. In this way, if a requested content is available in a requesting peer, then the content is downloaded from this peer. Client_thread_2 is responsible for receiving content from one or more peers in the system. To request contents from other peers, each peer uses the database with IP address and name of peers. A peer can receive multimedia contents from the main sources and from other requesting peers. These contents are stored in a database. Using this information each peer can work as server and redistribute information to other peers. Server_thread is responsible of this task in each peer.

4 Implementation

We put in practice our scheme by using different entities and servers, these entities are peers and servers. The servers store information about the connected peers and the file shared in each peer. There are different dedicated servers and each of them gives out and stores information about different types of files, which means there is a server for video another one for images another for music, another for pdf, and another one for any other kind of file so that we can do it multi-source. In this work, two main

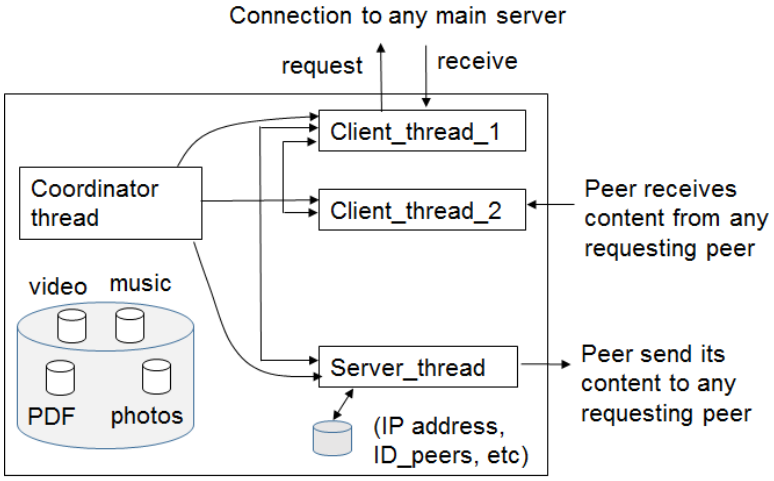


Fig. 2. Peer model.

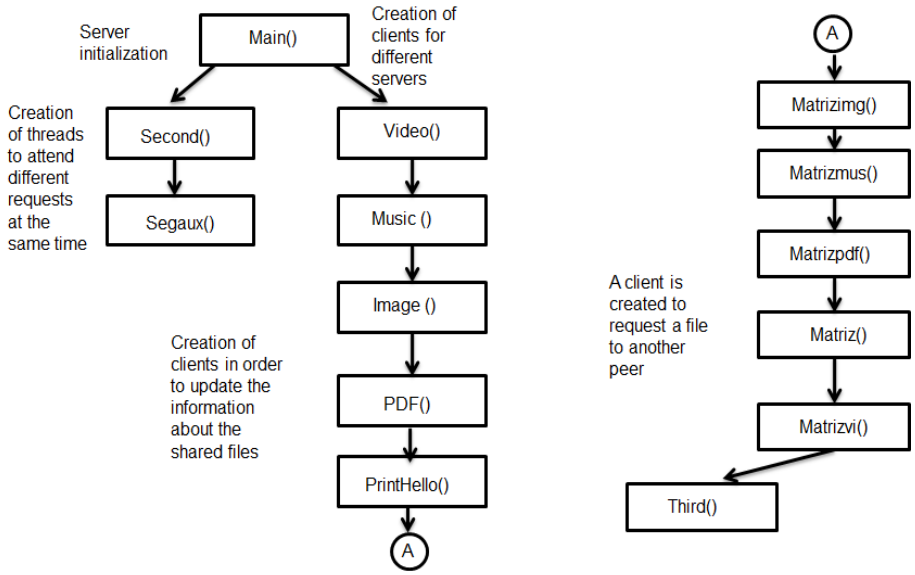


Fig. 3. Flow diagram for peer application.

applications called peer and server has been implemented. Each node in the P2P network runs a peer application, such that each node must receive and send files at the same time. To reach time goal, peer application performs both tasks simultaneously. Peer application is formed by two parts: a server and a client. Server part always is listening in order to attend to other peers, while client part makes different functions such as uploading files, display files and exit. Peer application is placed in each node of the P2P network (in each individual computer). A peer is an entity that sends and receives files at the same time. Peer application creates different threads when is running. First thread is to manipulate the server, and then other threads are created to connect them to the different servers. Each peer inserts its IP address, the amount of shared files with their names. Each server sends to all peers the information about the different connected peers. Each server sends updated information about the connected peer to all peers in the system. Figure 3 shows the flow diagram for peer application. Here, we can see the different steps developed by this application.

Our second application is the server. This application is responsible to give information about the peers in the system and store all files to be shared. A server is receiving requesting from peers while is sending files to them. While a server application is running, a socket is listening and waiting for new requests. When a new request from a peer is received in the server, a new thread is created to attend this request. Each server has a global matrix where IP address and the files names that peers want to share are registered. For each file to be shared by a server is created a thread toward that peer in order to store that file. Peer receives the matrix with information of connected peers and the shared files by these peers. An experimental prototype is implemented using five different servers. Each server offers a dedicated service, such that each server manages a specific type of file: video, music, images, PDF and one for other kind of files. Main steps for server application is shown in Figure 4. A matrix is

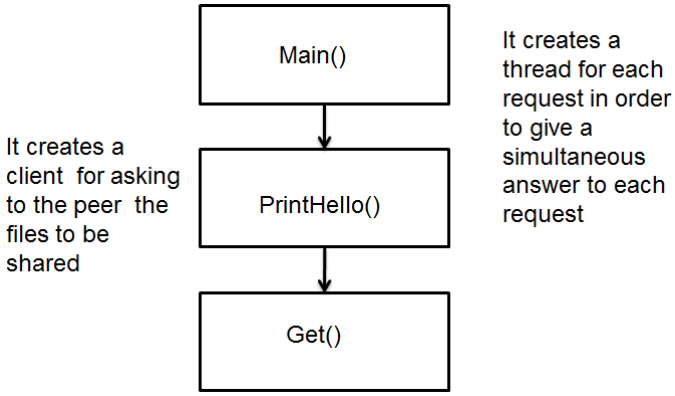


Fig. 4. Flow diagram for server application.

used to store the name of the shared files and the IP address of the peers in order to request files to different peers.

We have evaluated the operation of our prototype in a local red of our campus. Both servers and peers work correctly. Files are sent from each dedicated server and each requesting peer correctly, and after this, each peer can forward the received files to rest of peers. To compare performance of our collaborative P2P scheme against a distribution scheme based on client-server, we have measured the distribution time to requesting peers. Our work is in progress, and preliminary tests have been done. First, we distribute four files of 28MB to two clients from a server at same time. Client 1 receives the four files from the source in 7.56 minutes, while client 2 receives the four files from the source in 8.08 minutes. On the other hand, using P2P architecture, peer 1 receives the four requested files in 2.25 minutes, while peer 2 receives the four files in 2.16 minutes. Preliminary results show that P2P architecture presents a best performance than architecture based on client-server because. This is because the server is congested to send all files, while in the P2P scheme all nodes collaborate to distribute content faster. However, obtained measurements may change depending on the variation of the network. We can continue testing our collaborative transmission scheme in order to make more efficient our proposal.

5 Conclusions

There is currently a high demand for multimedia content, and collaboration among requesting nodes play an important role. In this paper has been proposed a collaborative multi-source scheme to distribute multimedia content to many requesting peers. Collaboration between peers is implemented by using a peer-to-peer network. Our framework is suited for collaborative environments, where the system inherently has multiple senders. Using this proposed approach, the sources can distribute their workload between all requesting peers, and the system can improve its performance. Our collaborative scheme uses threads to establish collaboration connections with other peers. The number of threads is variables and depends on the amount of established

connections. In each peer, a coordinator thread deals with the incoming requests and creates the rest of threads that handle the requests. Our current effort is focused to complete the implementation of our proposed framework for video streaming sessions.

References

1. Milojicic, D., Halogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rolling, S., Xu, Z.: Peer-to-Peer Computing. HP Labs Technical Report HPL, 57 (2002)
2. Cohen, B.: Incentives to build robustness in BitTorrent. Technical report, http://www.bittorrent. com/bittorrentecon.pdf (2003)
3. Pouwelse, A., Garbacki, P., Wang, J., Bakker, A., Yang, J., Iosup, A., Epema, D. H., Reinders, M., Van Steen, M. R., Sips, H.: TRIBLER: a social-based peer-to-peer system. Journal Concurrency and Computation: Practice & Experience, Vol. 20, No. 2, pp. 127–138 (2008)
4. The eMule Project: https://www.emule-project.net (2016)
5. Jannotti, J., Gifford, D. K., Johnson, K. L., Kaashoek, M. F., O'Toole Jr., J. W.: Overcast: Reliable Multicasting with an Overlay Network. In: Proc. of the 4th Symposium on Operating System Design and Implementation (OSDI'00), San Diego, CA, USA, pp. 197–212 (2000)
6. Chawathe, Y.: Scattercast: An Adaptable Broadcast Distribution Framework. In: ACM Multimedia Systems Journal Special Issue on Multimedia Distribution (2002)
7. Ripeanu, M., Foster, I., Iamnitchi, A.: Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design. Internet Computing Journal, Vol. 6 (2002)
8. Bharambe, A. R., Rao, S. G., Padmanabhan, V. N., Seshan, S., Zhang, H.: The Impact of Heterogeneous Bandwidth Constraints on DHT Based Multicast Protocols. In: Proc. of the 4th International Workshop IPTPS (2005)
9. Pucha, H., Andersen, D. G., Kaminsky, M.: Exploiting Similarity for Multi-Source Downloads Using File Handprints. In: Proc. of the 4th USENIX NSDI 07, Cambridge, MA, USA (2007)
10. Felber, P., Kermarrec, A. M., Leonini, L., Riviere, E., Voulgaris, S.: Pulp: An adaptive gossip-based dissemination protocol for multisource message streams. Peer-to-Peer Networking and Applications, Springer US (2012)
11. Cheng, B., Stein, L., Jin, H., Liao, X., Zhang, Z.: Gridcast: improving peer sharing for P2P VoD. In: ACM TOMCCAP (2008)
12. Dong, X. L., Berti-Equille, L., Srivastava, D.: Data fusion: resolving conflicts from multiple sources. Handbook of Data Quality, pp. 293–318 (2013)
13. López, F. A., Steinbach, E.: Multi-source video multicast in peer-to-peer networks. In: Proc. of the IEEE International Symposium on Parallel and Distributed Processing, Miami, FL, USA, pp. 1–8 (2008)
14. Sayit, M., Demirci, S., Kaymak, Y., Tunali, E.: Adaptive, incentive and scalable dynamic tree overlay for p2p live video streaming. In: Peer-to-Peer Networking and Applications, pp. 1–15 (2015)
15. Kuo, J. L., Shih, C. H., Ho, C. Y., Chen, Y. C.: Advanced bootstrap and adjusted bandwidth for content distribution in peer-to-peer live streaming. In: Peer-to-Peer Networking and Applications, pp 1–18 (2014)