

دراسة شبكات الند للند

من أجل البث الفيديوي

*Peer-to-Peer networks for
video streaming*

شكر

أتوجه بجزيل الشكر لمشرفي البحث، الأستاذ المساعد الدكتور محمد خالد شاهين من جامعة دمشق، لما قدمه لي من نصائح وتوجيهات ، ودعم علمي. فما كنت لأستطيع استكمال البحث لولا دعمه وتشجيعه.

وكذلك أتوجه بالشكر للجزيل للبروفسور جان لوكا ماتزيني من جامعة فيرارا الإيطالية، لما بذله من جهد لإنجاز هذا العمل، من خلال ملاحظاته وتوجيهاته القيّمة.

وأخيراً أتقدم بالشكر للجزيل إلى أعضاء لجنة الحكم الموقرين الذين رافقوني خلال مسيرتي التعليمية والذين أسهمت ملاحظاتهم وتوجيهاتهم في إنهاء هذا البحث.

إهداء

أهدي هذه الأطروحة لعائلتي التي لم تبخل علي يوماً بالدعم والتشجيع حتى استطعت إنجاز هذا العمل.

كما أهدي هذه الأطروحة لأصدقائي وزملائي لما قدموه لي من مساعدة وتشجيع خلال سنين الدراسة.

وأخيراً أهدي هذا البحث لكل من ساهم بإنجازه.

أعضاء لجنة الحكم والقرار الناظم بذلك

- د. عبد الرزاق البدوية الأستاذ في قسم هندسة الالكترونيات والاتصالات اختصاص نظم الكترونية عضواً
- د. محمد خالد شاهين الأستاذ المساعد في قسم هندسة الالكترونيات والاتصالات اختصاص اتصالات عضواً مشرفاً
- د. اسماعيل جنود الأستاذ المساعد في قسم هندسة الالكترونيات والاتصالات اختصاص معالجة الإشارة عضواً

بموجب قرار مجلس البحث العلمي والدراسات العليا رقم /437/ المتخذ بالجلسة / 4 /

تاريخ 2012/11/12.

فهرس المحتويات

9	List of Figures قائمة الأشكال
13	List of Tables قائمة الجداول
14	Glossary المصطلحات
14	المصطلحات مرتبة أبجدياً حسب اللغة الأنكليزية
23	المصطلحات مرتبة أبجدياً حسب اللغة العربية
33	Abbreviation الاختصارات
34	الملخص
35	المقدمة
37	1. إطار البحث
38	1.1. تمهيد البحث
39	2.1. الهدف من البحث
39	3.1. محددات البحث
40	4.1. الخلاصة
41	2. شبكات الند للند peer-to-peer networks
42	1.2. شبكة المخدم/زبون
43	2.2. شبكة الند للند
45	3.2. مزايا تطبيقات الند للند
45	1.3.2. التكلفة المنخفضة
45	2.3.2. الفعالية
46	3.3.2. المرونة
46	4.2. تحديات تطبيقات الند للند
46	1.4.2. مشاركة بيئة العمل
46	2.4.2. ديناميكية الشبكة
46	3.4.2. ديناميكية خصائص العقد

46	4.4.2. عدم تجانس الشبكة
46	5.2. تصنيف شبكات الند للند
47	1.5.2. الشبكة الصافية
48	2.5.2. الشبكة الهجينة
48	6.2. التشارك في الموارد
48	1.6.2. عرض المجال
48	2.6.2. موارد المعالجة
48	3.6.2. التخزين
49	7.2. الخلاصة
50	3. شبكات الند للند من أجل مشاركة الملفات
51	1.3. بعض تطبيقات الند للند الشهيرة
51	1.1.3. التطبيق نابستر
52	2.1.3. التطبيق غنوتلا
54	3.1.3. التطبيق كازا
56	4.1.3. التطبيق كورد
59	5.1.3. التطبيق بت تورنت
60	2.3. الخلاصة
61	4. شبكات الند للند من أجل البث الفيديوي
62	1.4. تقنيات التعامل مع الفيديو
62	2.4. تحديثات شبكات الند للند من أجل البث الفيديوي
63	3.4. تصنيف تطبيقات الند للند من أجل بث الفيديو
64	1.3.4. تصنيف تطبيقات الند للند لبث الفيديو اعتماداً على توضع العقد
65	2.3.4. تصنيف منظومات الند للند من أجل البث الفيديوي تبعاً لبنية المنظومة
66	4.4. بعض تطبيقات الند للند المستخدمة من أجل بث الفيديو
66	1.4.4. التطبيق نايس
67	2.4.4. التطبيق سبلت ستريم

68	3.4.4. التطبيق ناردا
68	4.4.4. التطبيق بوليت
68	5.4.4. التطبيق كوينت
69	6.4.4. التطبيق سبريد إت
69	7.4.4. التطبيق زيكا زاك
71	5.4. الخلاصة
73	5. نموذج التطبيق المقترح باستخدام شبكات الند للند
74	1.5. البنية المقترحة للمنظومة
75	2.5. آلية عمل التطبيق
76	3.5. مكونات المنظومة
76	1.3.5. المخدم المركزي
77	2.3.5. المصدر
78	3.3.5. المستخدم
79	4.5. آلية تنظيم العقد ضمن الشبكة
81	5.5. آلية تقسيم الملف وإعادة تجميعه
81	6.5. آلية إعادة بناء الشبكة
83	7.5. التنفيذ البرمجي للتطبيق
83	1.7.5. المكتبات البرمجية المستخدمة
84	2.7.5. التنفيذ البرمجي للمخدم
88	3.7.5. التنفيذ البرمجي للمصدر
94	4.7.5. التنفيذ البرمجي للمستخدم
99	8.5. الخلاصة
100	6. النتائج
101	1.6. إعداد بيئة الاختبار
102	2.6. المتوسطات الأساسية المستخدمة في اختبار التطبيق
102	3.6. مقارنة التطبيق مع بنية المخدم/زبون

103	4.6. أداء التطبيق من أجل مشاركة الملفات
105	1.4.6. أداء التطبيق من أجل أنواع اتصال مختلفة
106	2.4.6. دراسة مستويات التطبيق من أجل عرض مجال وصلة 128Kbps
107	3.4.6. دراسة مستويات التطبيق من أجل عرض مجال وصلة 384Kbps
108	4.4.6. دراسة مستويات التطبيق من أجل عرض مجال وصلة 1Mbps
109	5.4.6. دراسة مستويات التطبيق من أجل عرض مجال وصلة 2Mbps
110	5.6. اختبار أداء التطبيق من أجل البث الفيديوي
110	1.5.6. أداء التطبيق من أجل أنواع اتصال مختلفة
111	2.5.6. دراسة مستويات التطبيق من أجل عرض مجال وصلة 128Kbps
113	3.5.6. دراسة مستويات التطبيق من أجل عرض مجال وصلة 384Kbps
114	4.5.6. دراسة مستويات التطبيق من أجل عرض مجال وصلة 1Mbps
116	5.5.6. دراسة مستويات التطبيق من أجل عرض مجال وصلة 2Mbps
117	6.6. مقارنة التطبيق المقترح مع بعض التطبيقات الشهيرة
118	7.6. مناقشة النتائج
119	8.6. الخلاصة
120	7. الاستنتاجات والتوصيات
121	1.7. ملخص البحث
121	2.7. مخرجات البحث research outputs
122	3.7. الخلاصة
122	4.7. المقترحات والتوصيات
124	8. المراجع
128	9. الملحقات
129	أ. برمجة المقبس socket programming
135	ب. FFMPEG
143	ج. SDL
148	د. البرمجة بلغة سي
159	هـ. بعض تطبيقات الند للند الشهيرة

قائمة الأشكال List of Figures

الشكل	الصفحة
الشكل (1-2) مثال على بنية المخدم/زبون	42
الشكل (2-2) مثال على بنية شبكة الند للند	43
الشكل (3-2) تصنيف شبكات الند للند	47
الشكل (1-3) بنية التطبيق نابستر	51
الشكل (2-3) آلية عمل التطبيق نابستر	52
الشكل (3-3) مراحل عمل التطبيق غنوتلا	54
الشكل (4-3) بنية التطبيق كازا	55
الشكل (5-3) آلية عمل التطبيق كازا	56
الشكل (6-3) توضع العقد والمفاتيح على حلقة كورد	57
الشكل (7-3) البحث بالطريقة البسيطة ضمن حلقة كورد	58
الشكل (8-3) البحث باستخدام الطريقة المتصعدة	58
الشكل (9-3) التطبيق بت تورنت	59
الشكل (1-4) تصنيف تطبيقات الند للند من أجل بث الفيديو	63
الشكل (2-4) بنية التطبيق نايس	67
الشكل (3-4) المبدأ الأساسي لعمل التطبيق سبلت ستريم	67
الشكل (4-4) التنظيم الإداري للعقد	70
الشكل (5-4) الشجرة متعددة البث المبنية على أساس التنظيم الإداري	71
الشكل (1-5) البنية المقترحة للتطبيق	74
الشكل (2-5) آلية عمل التطبيق المقترح	76
الشكل (3-5) بنية قاعدة البيانات في المخدم	77
الشكل (4-5) بنية جدول التخصيص في المخدم	77
الشكل (5-5) تخصيص حيزات الوصلة من أجل حيزات المحتوى	78
الشكل (6-5) المخطط التدفقي لآلية بناء الشبكة	80

- الشكل (5-7) بنية الحاوية المقترحة 81
- الشكل (5-8) آلية البحث عن العقدة الفاشلة 82
- الشكل (5-9) المخطط التدفقي لإعادة بناء الشبكة عند فشل عقدة ضمنها 82
- الشكل (5-10) آلية تهيئة الاتصال 84
- الشكل (5-11) الخطوات البرمجية لتهيئة الاتصال 84
- الشكل (5-12) المخطط التدفقي من أجل التعامل مع المصدر 85
- الشكل (5-13) برنامج سي من أجل التعامل مع المصدر 85
- الشكل (5-14) المخطط التدفقي لعمل المخدم عند ورود طلب من المستخدم 86
- الشكل (5-15) برنامج سي من أجل التعامل مع المستخدم 87
- الشكل (5-16) برنامج سي المسؤول عن إعادة بناء الشبكة عند فشل أحد العقد 88
- الشكل (5-17) برنامج سي المسؤول عن اتصال المصدر مع المخدم 89
- الشكل (5-18) برنامج سي المسؤول عن استقبال طلبات المستخدمين 90
- الشكل (5-19) برنامج سي المستخدم لإيجاد معلومات الترميز وإرسالها إلى المستخدم 91
- الشكل (5-20) برنامج سي المسؤول عن تهيئة الحاوية 92
- الشكل (5-21) برنامج سي المسؤول عن قراءة رزم الفيديو ووضعها ضمن الحاوية 93
- الشكل (5-22) برنامج سي المسؤول عن وضع الحاوية في المكان المخصص ضمن الوصلة 94
- الشكل (5-23) برنامج سي المسؤول عن الاتصال بين المستخدم والمخدم 95
- الشكل (5-24) برنامج سي المكتوب من أجل الاتصال بالمصدر 95
- الشكل (5-25) برنامج سي المكتوب من أجل استقبال البيانات من المصدر 96
- الشكل (5-26) آلية التعامل مع الفيديو 97
- الشكل (5-27) برنامج سي لتهيئة الشاشة من أجل عرض الفيديو 97
- الشكل (5-28) برنامج سي المكتوب من أجل عرض الفيديو على الشاشة 98
- الشكل (5-29) برنامج سي من أجل إعادة الإرسال 99
- الشكل (6-1) مقارنة عدد مستخدمي تطبيق المخدم/زبون وتطبيق الند للند المقترح 103
- الشكل (6-2) مخطط اتصال العقد بحالة المخدم/زبون وتطبيق الند للند المقترح 103
- الشكل (6-3) مخطط الشبكة من أجل عرض مجال وصلة مختلف 104

- الشكل (4-6) مخطط الشبكة من أجل قيم عرض مجال وصلة هابطة مختلفة 104
- الشكل (5-6) مخطط الشبكة من أجل عدد عقد مختلف 105
- الشكل (6-6) علاقة حجم الملف بزمان الاستقبال من أجل قيم مختلفة لـ BWL 105
- الشكل (7-6) العلاقة بين زمن الاستقبال وحجم الملفات من أجل BWL=128Kbps 106
- الشكل (8-6) العلاقة بين عدد المستخدمين وعرض مجال الوصلة الهابطة 106
- الشكل (9-6) العلاقة بين زمن الاستقبال وحجم الملف من أجل عدد مختلف للعقد (128Kbps) 107
- الشكل (10-6) العلاقة بين زمن الاستقبال وحجم الملفات من أجل BWL=384Kbps 107
- الشكل (11-6) العلاقة بين زمن الاستقبال وحجم الملف من أجل عدد مختلف للعقد (384Kbps) 108
- الشكل (12-6) العلاقة بين زمن الاستقبال وحجم الملفات من أجل BWL=1Mbps 108
- الشكل (13-6) العلاقة بين زمن الاستقبال وحجم الملف من أجل عدد مختلف للعقد (1Mbps) ... 109
- الشكل (14-6) العلاقة بين زمن الاستقبال وحجم الملفات من أجل BWL=2Mbps 109
- الشكل (15-6) العلاقة بين زمن الاستقبال وحجم الملف من أجل عدد مختلف للعقد (2Mbps) ... 110
- الشكل (16-6) علاقة حجم الفيديو بزمان الاستقبال من أجل قيم مختلفة لـ BWL 111
- الشكل (17-6) علاقة حجم الفيديو بزمان التأخير من أجل قيم مختلفة لـ BWL 111
- الشكل (18-6) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل BWL=128Kbps 111
- الشكل (19-6) العلاقة بين زمن التأخير وحجم الفيديو من أجل BWL=128Kbps 112
- الشكل (20-6) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل عدد مختلف للعقد (128Kbps) 112
- الشكل (21-6) العلاقة بين زمن التأخير وحجم الفيديو من أجل عدد مختلف للعقد (128Kbps) .. 112
- الشكل (22-6) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل BWL=384Kbps 113
- الشكل (23-6) العلاقة بين زمن التأخير وحجم الفيديو من أجل BWL=384Kbps 113
- الشكل (24-6) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل عدد مختلف للعقد (384Kbps) 114
- الشكل (25-6) العلاقة بين زمن التأخير وحجم الفيديو من أجل عدد مختلف للعقد (384Kbps) .. 114
- الشكل (26-6) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل BWL=1Mbps 114
- الشكل (27-6) العلاقة بين زمن التأخير وحجم الفيديو من أجل BWL=1Mbps 115
- الشكل (28-6) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل عدد مختلف للعقد (1Mbps) 115
- الشكل (29-6) العلاقة بين زمن التأخير وحجم الفيديو من أجل عدد مختلف للعقد (1Mbps) 115

الشكل (6-30) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل $BWL=2Mbps$	116
الشكل (6-31) العلاقة بين زمن التأخير وحجم الفيديو من أجل $BWL=2Mbps$	116
الشكل (6-32) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل عدد مختلف للعقد (2Mbps).....	117
الشكل (6-33) العلاقة بين زمن التأخير وحجم الفيديو من أجل عدد مختلف للعقد (2Mbps).....	117
الشكل (أ-1) آلية الربط التي يقدمها المقبس	130
الشكل (أ-2) آلية برمجة المقبس من أجل بروتوكول بنية التحكم في الإرسال	131
الشكل (أ-3) آلية برمجة المقبس من أجل بروتوكول برقية معطيات المستخدم	131
الشكل (ب-1) المخطط التدفقي لتصميم تطبيق فيديو أو صوت	136
الشكل (ه-1) واجهة تطبيق ميو تورنت μ torrent	160
الشكل (ه-2) واجهة التطبيق غنوتلا تربو Gnutella Turbo	161
الشكل (ه-3) واجهة التطبيق فروست واير FrostWire	162

قائمة الجداول List of Tables

الجدول	الصفحة
الجدول (1-2) مقارنة بين شبكات الند للند وشبكات المخدم/زبون	44
الجدول (1-3) أنماط الرسائل في بروتوكول غنوتلا	52
الجدول (1-5) مقارنة بين التطبيق المقترح وبعض التطبيقات الشهيرة	75
الجدول (2-5) المكتبات البرمجية المستخدمة	83
الجدول (1-6) المتوسطات المستخدمة في اختبار التطبيق	102
الجدول (2-6) مقارنة تطبيق الند للند المقترح مع بيئة المخدم/زبون (عدد المستخدمين)	102
الجدول (3-6) مقارنة تطبيق الند للند المقترح مع بيئة المخدم/زبون (زمن الاستقبال)	102
الجدول (4-6) مقارنة بين التطبيق المقترح وبعض التطبيقات	117
الجدول (أ-1) الفرق بين نوعي المقابس	117
الجدول (د-1) أنواع البيانات المستخدمة بلغة سي C مع عدد البايتات التي يشغلها كل نوع	117
الجدول (د-2) الرموز المستخدمة مع تابع الطباعة	151
الجدول (د-3) الرموز المستخدمة في تابع الطباعة من أجل طباعة المتغيرات على الشاشة	151
الجدول (د-4) أنواع الحلقات وعمل كل منها	152
الجدول (د-5) أهم التوابع المستخدمة مع السلاسل المحرفية	154
الجدول (د-6) أنماط التعامل مع الملفات	156

Glossary المصطلحات

المصطلحات مرتبة أبجدياً حسب اللغة الانكليزية

A	
Accept	قبول
Adjustment	ضبط
Administrative organization	تنظيم إداري
Allocation table	جدول التخصيص
Appending	إلحاق
Assigning	تخصيص
Associated head	رأس مساعد
Availability	المتاحة
B	
Bandwidth	عرض مجال
BitTorrent	بت تورنت
Bottleneck	عنق الزجاجة
Broadcasting	البث العام
Branch	تفرع
Buffer	ذاكرة مؤقتة
Bullet	بوليت
C	
Capabilities	قدرات
Central server based architecture	بنية ذات المخدم المركزي
Centralized directory	دليل مركزي

Centralized indexing	فهرسة مركزية
ChainSaw	تشين ساو
Chord	كورد
Chord ring	حلقة كورد
Chunk	كتلة
Client	زبون
Client/Server network	شبكة مخدم/زبون
Cluster	عنقود
Connection	اتصال
Connection oriented	اتصال موجه
Connectivity	قابلية الاتصال
Container	حاوية
Content	محتوى
CoopNet	كوبنت
D	
Database	قاعدة معطيات
Decentralized indexing	فهرسة موزعة
Decrement	نقصان
Degree of centralized	درجة المركزية
Delay	تأخير
Delivery	إيصال
Digital TV/HDTV	البث التلفزيوني الرقمي عالي الجودة
Distributed based architecture	بنية موزعة
Distributed computing	معالجة موزعة
Donet/CoolStreaming	التطبيق دونت/كول ستريمينغ

Downlink	وصلة هابطة
Download	إتيان
E	
E-learning	التعلم الإلكتروني
Entity	مكون
F	
Fail node	عقدة فاشلة
Fault-tolerant	مناعة ضد الفشل
Fetch	جلب
File sharing	مشاركة الملفات
Finger table	جدول الدلالة
Firewall	جدار النار (الحماية)
Flexible naming	تسمية مرنة
Flooding	تعويم
Foreign cluster	عنقود خارجي
Foreign head	رأس خارجي
Foreign subordinate	مرؤوس خارجي
Function	تابع
G	
Global knowledge	معرفة عامة
Gnutella	التطبيق غنوتلا
Google	غوغل
H	
Head	رأس
Heterogeneity	عدم تجانس

Heterogeneous	غير متجانس
Hexadecimal	النظام السداسي عشري
Hierarchical based architecture	بنية هرمية
Host name resolution	الربط بين العناوين وأسماء المضيفين
Hybrid	هجين
I	
Identifier	معيّن
Indexing	فهرسة
In-order	ترتيب
Instant messaging	تبادل الرسائل الفورية
Instructions	تعليمات
Internet access	النفاد إلى الانترنت
Internet conferencing	المؤتمرات عبر الانترنت
K	
KaZaA	التطبيق كازا
Key location	توضع المفتاح
L	
Link	وصلة
Linux	نظام اللينوكس
Listening	استماع
Load balance	موازنة الحمل
Local area network	شبكة محلية
Local knowledge	معرفة محلية
Logical	منطقي
Loop	حلقة

M	
Maintenance	صيانة
Manipulated	معالجة
Mesh-based systems	منظومات شبكية
Message digest	أرقام الرسالة
Metadata	معطيات مترفّعة
Multicast	متعدد الوجهات
Multicast tree	شجرة البث المتعدد
Multi-layer hierarchy	بنية هرمية متعددة الطبقات
Multimedia streaming	بث الوسائط المتعددة
N	
Napster	التطبيق نابستر
Narada	التطبيق ناردا
Network layer	طبقة الشبكة
NICE	التطبيق نايس
Node	عقدة
Non head	غير رئيسي
Non static	غير ساكن
Notification	إعلام مسبق
O	
Octal	النظام الثماني
Operators	مؤثرات
Ordinary node	عقد عادية
Overcast	التطبيق أوفر كاست
Overhead	معلومات مضافة

Overlay	بنية تراكبية
Overload	تحميل زائد
P	
Packet	رزمة معطيات
PALS	التطبيق بالس
Pause	إيقاف مؤقت
PeerCast	التطبيق بيركاست
Peer-to-peer network	شبكة الند للند
Ping	أخذ
Play	تشغيل
Play back	زمن التأخير
Pointer	مؤشر
Pong	ردّ
Port	منفذ
Portable	متنقلة
PRIME	التطبيق برايم
Procedure	إجراء
PROMISE	التطبيق برومايس
Protocol	بروتوكول
Publish	نشر
Pure	صافي
Push	دفع
Q	
Quality	جودة
Query	استفسار

Query Hit	إصابة الاستفسار
R	
Real-time	الزمن الحقيقي
Reference	مرجع
Relay	مُرَحِّل
Reliability	وثوقية
Reliable	موثوق
Rendezvous point	نقطة تجميع
Resource	مورد
Resource sharing	مشاركة الموارد
Responsibilities	مسؤوليات
Robustness	متانة
Routing table	جدول توجيه
S	
Scalability	تصعد
Scalable key location	الطريقة المتصعدة لتوضع المفتاح
Scalar	مقيس
Search	بحث
Secure Hash Algorithm	خوارزمية التشفير السري
Security	أمن
Seeking	نقطة البدء بالعرض
Sending time	زمن الإرسال
Sequence number	رقم تسلسلي
Server	مخدم
Service	خدمة

Shortest Widest Path Algorithm	خوارزمية المسار الأعرض والأقصر
Simple key location	الطريقة البسيطة لتوضع المفتاح
Single point of failure	نقطة فشل واحدة
Slot	حيز
Slot allocation	تخصيص الحيزات
Socket	مقبس
Source	مصدر
SplitStream	التطبيق سبلت ستريم
SpreadIT	التطبيق سبريد إت
Standard	معايير قياسية
Static	ساكن
Storage	تخزين
Stream	دفق
Streaming server	مخدم بث
Structured	بنوي
Subordinate	مرؤوس
Sub-stream	جزء
Super node	عقد فائقة
Swarming	حشد
Switch	مبدل
Symbol	رمز
Symmetric	متناظر
System	منظومة

T	
Terrestrial	خطوط النقل الأرضية
Time to live	زمن الحياة
Timestamp	معلومات زمنية
Tracker	متعقب
Transmission control protocol/Internet protocol	بروتوكول بنية التحكم في الإرسال/الترباط الشبكي
Tree-based system	منظومة شجرية
U	
Unsaturated	غير مشبع
Unstructured	غير بنيوي
Update	تحديث
Uplink	وصلة صاعدة
Upload	تحميل
User datagram protocol/Internet protocol	بروتوكول برقية معطيات المستخدم/الترباط الشبكي
V	
Video conferencing	المؤتمرات الفيديوية
Video On Demand	الفيديو عند الطلب
Video streaming	البث الفيديوي
W	
Well-known	محددة
World Wide Web	النسيج العنكبوتي العالمي
Z	
ZigZag	التطبيق زيك زاك

المصطلحات مرتبة أبجدياً حسب اللغة العربية

أ	
Connection	اتصال
Connection oriented	اتصال موجه
Download	إتيان
Procedure	إجراء
Ping	أخذ
Message digest	أرقام الرسالة
Query	استفسار
Listening	استماع
Query Hit	إصابة الاستفسار
Notification	إعلام مسبق
Appending	إلحاق
Security	أمن
Delivery	إيصال
Pause	إيقاف مؤقت
ب	
Digital TV/HDTV	البث التلفزيوني الرقمي عالي الجودة
Broadcasting	البث العام
Video streaming	البث الفيديوي
Multimedia streaming	بث الوسائط المتعددة
Search	بحث
Protocol	بروتوكول
User diagram protocol/Internet protocol	بروتوكول برقية معطيات المستخدم/الترايط

	الشبكي
Transmission control protocol/Internet protocol	بروتوكول بنية التحكم في الإرسال/الترابط الشبكي
Overlay	بنية تراكبية
Central server based architecture	بنية ذات المخدم المركزي
Distributed based architecture	بنية موزعة
Hierarchical based architecture	بنية هرمية
Multi-layer hierarchy	بنية هرمية متعددة الطبقات
Structured	بنوي
ت	
Function	تابع
Delay	تأخير
Instant messaging	تبادل الرسائل الفورية
Update	تحديث
Upload	تحميل
Overload	تحميل زائد
Storage	تخزين
Assigning	تخصيص
Slot allocation	تخصيص الحيزات
In-order	ترتيب
Flexible naming	تسمية مرنة
Play	تشغيل
Forward error correction	تصحيح الخطأ الأمامي
Scalability	تصعد

Overcast	التطبيق أوفركاست
PALS	التطبيق باليس
BitTorrent	التطبيق بت تورنت
PRIME	التطبيق برايم
PROMISE	التطبيق برومايس
Bullet	التطبيق بوليت
PeerCast	التطبيق بيركاست
ChainSaw	التطبيق تشين ساو
Donet/CoolStreaming	التطبيق دونت/كول ستريمينغ
ZigZag	التطبيق زيك زاك
SpreadIT	التطبيق سبريد إيت
SplitStream	التطبيق سبلت ستريم
Gnutella	التطبيق غنوتلا
KaZaA	التطبيق كازا
CoopNet	التطبيق كوبنت
Chord	التطبيق كورد
Napster	التطبيق نابستر
Narada	التطبيق ناردا
NICE	التطبيق نايس
E-learning	التعلم الإلكتروني
Flooding	تعويم
Brunch	تفريع
Administrative organization	تنظيم إداري
Key location	توضع المفتاح

ث	
Bi-directional	ثنائي الاتجاه
ج	
Firewall	جدار النار (الحماية)
Allocation table	جدول التخصيص
Finger table	جدول الدلالة
Routing table	جدول توجيه
Sub-stream	جزء
Fetch	جلب
Quality	جودة
ح	
Container	حاوية
Swarming	حشد
Loop	حلقة
Chord ring	حلقة كورد
Slot	حيز
خ	
Service	خدمة
Terrestrial	خطوط النقل الأرضية
Secure Hash Algorithm	خوارزمية التشفير السري
Shortest Widest Path Algorithm	خوارزمية المسار الأعرض والأقصر
د	
Degree of centralized	درجة المركزية
Push	دفع

Stream	دفق
Centralized directory	دليل مركزي
ذ	
Buffer	ذاكرة مؤقتة
ر	
Head	رأس
Foreign head	رأس خارجي
Associated head	رأس مساعد
Host name resolution	الربط بين العناوين وأسماء المضيفين
Pong	ردّ
Packet	رزمة معطيات
Sequence number	رقم تسلسلي
Symbol	رمز
ز	
Client	زبون
Sending time	زمن الإرسال
Play back	زمن التأخير
Real-time	الزمن الحقيقي
Time to live	زمن الحياة
س	
Static	ساكن
ش	
Peer-to-peer network	شبكة الند للند
Client/Server network	شبكة مخدم/زبون

Multicast tree	شجرة البث المتعدد
ص	
Pure	صافي
Maintenance	صيانة
ض	
Adjustment	ضبط
Data loss	ضياع المعطيات
ط	
Simple key location	الطريقة البسيطة لتوضع المفتاح
Scalable key location	الطريقة المتصاعدة لتوضع المفتاح
ع	
Heterogeneity	عدم تجانس
Bandwidth	عرض مجال
Node	عقدة
Ordinary node	عقدة عادية
Fail node	عقدة فاشلة
Super node	عقدة فائقة
Bottleneck	عنق الزجاجة
Cluster	عنقود
Foreign cluster	عنقود خارجي
غ	
Google	غوغل
Unstructured	غير بنيوي
Non head	غير رئيسي

Non static	غير ساكن
Heterogeneous	غير متجانس
Asymmetric	غير متناظر
Unsaturated	غير مشبع
ف	
Interval	فارق زمني
Indexing	فهرسة
Centralized indexing	فهرسة مركزية
Decentralized indexing	فهرسة موزعة
Video On Demand	الفيديو عند الطلب
ق	
Connectivity	قابلية الاتصال
Database	قاعدة معطيات
Accept	قبول
Capabilities	قدرات
ك	
Chunk	كتلة
م	
Switch	مبدل
Availability	المتاحة
Robustness	متانة
Multicast	متعدد الوجهات
Tracker	متعقب
Symmetric	متناظر

Content	محتوى
Well-known	محددة
Server	مخدم
Streaming server	مخدم بث
Reference	مرجع
Relay	مُرْجِل
Subordinate	مرؤوس
Foreign subordinate	مرؤوس خارجي
Responsibilities	مسؤوليات
File sharing	مشاركة الملفات
Resource sharing	مشاركة الموارد
Source	مصدر
Manipulated	معالجة
Distributed computing	معالجة موزعة
Global knowledge	معرفة عامة
Local knowledge	معرفة محلية
Metadata	معطيات مترفّعة
Timestamp	معلومات زمنية
Overhead	معلومات مضافة
Identifier	معيّن
Socket	مقبس
Scalar	مقيس
Entity	مكون
Fault-tolerant	مناعة ضد الفشل

Logical	منطقي
Mesh-based systems	منظومات شبكية
System	منظومة
Tree-based system	منظومة شجرية
Port	منفذ
Load balance	موازنة الحمل
Video conferencing	المؤتمرات الفيديوية
Internet conferencing	المؤتمرات عبر الانترنت
Operators	مؤثرات
Reliable	موثوق
Resource	مورد
Pointer	مؤشر
Descriptor	موصّف
ن	
World Wide Web	النسيج العنكبوتي العالمي
Publish	نشر
Octal	النظام الثماني
Hexadecimal	النظام السداسي عشري
Linux	نظام اللينوكس
Internet access	النفاد إلى الانترنت
Seeking	نقطة البدء بالعرض
Rendezvous point	نقطة تجميع
Single point of failure	نقطة فشل واحدة

هـ	
Hybrid	هجين
و	
Reliability	وثوقية
Link	وصلة
Uplink	وصلة صاعدة
Downlink	وصلة هابطة

الاختصارات Abbreviation

B	
BW	Bandwidth
Bunit	Bandwidth unit
BWu	Uplink Bandwidth
BWd	Downlink Bandwidth
C	
C/S	Client/Server
H	
HDTV	High Definition TV
I	
IP	Internet Protocol
M	
MDC	Multi Description Coding
P	
P2P	Peer-to-Peer
S	
SHA	Secure Hash Algorithm
T	
TCP	Transport Control Protocol
TL	Total number of Link slot
TS	Total number of Content slot
TTL	Time To Live
U	
UDP	User Datagram Protocol
V	
VOD	Video On Demand
W	
WWW	World Wide Web

المخلص

تعتمد شبكات الند للند (P2P) على التعاون بين المستخدمين والتشارك في الموارد. وقد شاع استخدام هذا النوع من الشبكات في الكثير من التطبيقات، من أشهرها مشاركة الملفات والبث الفيديوي. تقسم تطبيقات مشاركة الملفات، المعتمدة على بنية الند للند، إلى منظومات مركزية، تعتمد على وجود مخدم مركزي لتنظيم العمل، ومنظومات موزعة، كما يمكن أن تقسم إلى منظومات بنىوية أو غير بنىوية، تبعاً لآلية إدارة المعلومات وتنظيمها ضمن المنظومة.

على خلاف تطبيقات مشاركة الملفات، يجري في تطبيقات البث الفيديوي بدء عرض الفيديو مباشرة بعد استقبال جزء صغير من الفيديو فقط، ولا نحتاج إلى انتظار استقبال كامل الملف قبل فتحه أو تعديله.

يقدم هذا البحث تطبيق ند لند هجين يُستخدم من أجل مشاركة الملفات، وكذلك للبث الفيديوي، مُنفذ برمجياً باستخدام لغة السي C، كما يتضمن هذا البحث مجموعة اختبارات أُجريت على هذا التطبيق، والتي أكدت مزايا شبكات الند للند من حيث التصعد الكبير (زيادة عدد المستخدمين بدون الحاجة لتغيير في بنية التطبيق أو مكوناته) ووثوقية التطبيق.

المقدمة

أدى التطور التقني السريع، وظهور تقنيات نفاذ إلى الانترنت Internet access ذات سرعات عالية وعرض مجال bandwidth واسع، إلى ازدياد الطلب على تطبيقات مشاركة الملفات، وكذلك خدمات بث الوسائط المتعددة multimedia streaming. وقد اعتمدت هذه التطبيقات بادئ الأمر على بنية المخدم/زبون client/server والتي تقوم على وجود مخدم مركزي يخزن كافة المعطيات المراد التشارك بها، ويقوم بتلبية طلبات جميع المستخدمين. ولذلك كان من الضروري استخدام مخدمات بمساحة تخزين كبيرة، وعرض مجال واسع، ووثوقية عالية، ومن ثم تكلفة مرتفعة.

ومع ازدياد عدد مستخدمي تطبيقات مشاركة الملفات والبث الفيديوي، ظهرت عدم فاعلية هذه البنية، مما دفع إلى استخدام تقنيات البث متعدد الوجهات multicast، رغم ازدياد التكلفة والتعقيد. أدى ظهور شبكات الند للند وما تتمتع به من إمكانية التصعد scalability إلى حد كبير، وبتكلفة منخفضة جداً، إلى تركيز اهتمام مطوري تطبيقات الانترنت على هذه البنية. وتعتمد بنية الند للند على وجود عدد من العقد التي تتشارك في الموارد resources من أجل تبادل المعلومات، حيث تلعب كل عقدة دور المخدم والزبون بنفس الوقت. وتستقبل كل عقدة، ضمن بنية الند للند، المعلومات من عقدة أو أكثر، ثم تقوم بإعادة إرسالها إلى عقد أخرى.

استخدمت شبكات الند للند بادئ الأمر من أجل مشاركة الملفات عبر الانترنت، وأدت النتائج المشجعة التي حققتها شبكات الند للند إلى جذب نظر مطوري تطبيقات البث الفيديوي، وهكذا تركزت الأبحاث بشكل أساسي على تطوير شبكات الند للند وتنفيذها لملائمة متطلبات البث الفيديوي من حيث إيصال الفيديو إلى عدد كبير من المستخدمين وبأقل تأخير ممكن.

يهدف هذا البحث إلى تصميم تطبيق ند لند واستخدامه من أجل مشاركة الملفات، وكذلك للبث الفيديوي بالزمن الحقيقي real-time للاستفادة من مزايا هذه البنية، وتقييم الأداء من حيث التأخير delay.

جرى بناء التطبيق بالاعتماد على البنية الهرمية لشبكات الند للند، والتي تحتوي على مخدم مركزي مسؤول عن تنظيم العقد، وتخزين الملفات ضمن الشبكة. تمتاز هذه البنية بكفاءة عملية البحث عن الملفات وسرعتها، كما تمتاز بسهولة تنظيم العقد وإعادة بناء الشبكة بسرعة في حالات الفشل.

لبعض العقد. اعتمدنا في هذا التطبيق على تقسيم الملفات إلى أجزاء متساوية بالحجم (كتل) chunk، حيث يجري استقبال أجزاء الملف من عدة عقد، ومن ثم إعادة ترتيبها وتجميعها في المستقبل. يساعد تقسيم الملف إلى أجزاء على الاستغلال الأمثل للموارد وتقليل زمن الاستقبال. قمنا بتنفيذ التطبيق بشكل برمجي باستخدام لغة السي، ودراسة الأداء من حيث زمن الاستقبال وزمن التأخير play back، وهو الزمن اللازم قبل بدء عرض ملف الفيديو في حالة البث الفيديوي. كما قمنا بمقارنة التطبيق مع بيئة المخدم/زبون من حيث التصعد وزمن الاستقبال. تتضمن الأطروحة سبعة فصول مرتبة على الشكل الآتي:

يشتمل الفصل الأول على مقدمة عامة مؤلفة من تمهيد، والهدف من البحث، ومحددات البحث. ويبحث الفصل الثاني في بنية شبكات الند للند، وعرضاً لأهم مزاياها ومساوئها. في حين يتناول الفصل الثالث تطبيقات الند للند من أجل مشاركة الملفات بأنواعها المختلفة، مع بعض الأمثلة العملية عليها.

ويعالج الفصل الرابع استخدام شبكات الند للند في بث الفيديو، فيعرض للتحديات وطرق التصنيف وأهم التطبيقات المستخدمة.

في حين يعرض الفصل الخامس النموذج المقترح لتطبيق ند للند من أجل مشاركة الملفات وكذلك للبث الفيديوي، ويتضمن توضيحاً لبنية التطبيق وآلية تنظيم العقد وتقسيم الملف وإعادة تجميعه، وكذلك آلية إعادة بناء الشبكة في حال مغادرة العقد بشكل مفاجئ.

ويتناول الفصل السادس نتائج تنفيذ التطبيق بحالتين هما الفتح بعد الإتيان بالملف open after download، والتشغيل أثناء الإتيان بالملف play while downloading. وذلك باستخدام المحاكاة.

في الفصل السابع والأخير تحدثنا عن الاستنتاجات التي توصلنا إليها من خلال اختبار التطبيق وكذلك التوصيات المقترحة لتطوير التطبيق وتحسين أدائه.

تتضمن الأطروحة إضافةً إلى ذلك جداول المصطلحات والاختصارات المستخدمة بالبحث وكذلك لائحة المراجع المستخدمة، وملاحق.

إطار البحث

1.1. تمهيد البحث

أدى التطور التقني السريع، وظهور تقنيات نفاذ إلى الانترنت ذات سرعات عالية وعرض مجال واسع، إلى ازدياد الطلب على تطبيقات مشاركة الملفات، وكذلك خدمات بث الوسائط المتعددة مثل: الفيديو عند الطلب (VOD) Video On Demand، والتعلم الإلكتروني E-learning، والمؤتمرات الفيديوية video conferencing، والبث الفيديوي video streaming [1-4].

وقد اعتمدت هذه التطبيقات بادئ الأمر على بنية المخدم/زبون client/server، والتي تقوم على وجود مخدم مركزي يخزن كافة المعطيات المراد التشارك بها، ويقوم بتلبية طلبات جميع المستخدمين. ولذلك كان من الضروري استخدام مخدمات بمساحة تخزين كبيرة، وعرض مجال واسع، ووثوقية عالية، ومن ثم تكلفة مرتفعة [2، 5].

ومع ازدياد عدد مستخدمي تطبيقات مشاركة الملفات والبث الفيديوي، ظهرت عدم فاعلية هذه البنية، مما دفع إلى استخدام تقنيات البث متعدد الوجهات، رغم ازدياد التكلفة والتعقيد [6].

أدى ظهور شبكات الند للند وما تتمتع به من إمكانية التصعد إلى حد كبير، وبتكلفة منخفضة جداً، إلى تركيز اهتمام مطوري تطبيقات الانترنت على هذه البنية. وتعتمد بنية الند للند على وجود عدد من العقد التي تتشارك في الموارد من أجل تبادل المعلومات، حيث تلعب كل عقدة دور المخدم والزبون بنفس الوقت. وتستقبل كل عقدة، ضمن بنية الند للند، المعلومات من عقدة أو أكثر، ثم تقوم بإعادة إرسالها إلى عقد أخرى [7-9].

واستخدمت شبكات الند للند بادئ الأمر من أجل مشاركة الملفات عبر الانترنت، حيث ظهرت العديد من التطبيقات مثل تطبيق نابستر Napster المستخدم لتبادل ملفات الموسيقى، وتطبيق بت تورنت BitTorrent الذي يُعدّ من أشهر تطبيقات الند للند المستخدمة من أجل الإتيان بالملفات بسرعة وكفاءة [10، 11].

أدت النتائج المشجعة التي حققتها شبكات الند للند إلى جذب نظر مطوري تطبيقات البث الفيديوي، وهكذا تركزت الأبحاث بشكل أساسي على تطوير شبكات الند للند وتنفيذها لملائمة متطلبات البث الفيديوي من حيث إيصال الفيديو إلى عدد كبير من المستخدمين وبأقل تأخير ممكن [12، 13].

تقسم شبكات الند للند من أجل مشاركة الملفات إلى شبكات صافية، وشبكات هجينة تعتمد على وجود مخدم مركزي مسؤول عن تنظيم العقد والملفات ضمنها. يساعد وجود المخدم المركزي على تسهيل التحكم في بناء الشبكة وإدارتها، وكذلك تحسين فعالية البحث عن الملفات وتسريعها [14]. أما شبكات الند للند من أجل البث الفيديوي فتقسم إلى نوعين أساسيين هما المنظومات الشجرية tree-based وتتألف من عدد من الأشجار المستخدمة لإيصال كتل الفيديو، والمنظومات الشبكية mesh-based وفيها تتصل العقد مع بعضها البعض بشكل عشوائي [15].

يمتاز النمط الشجري بتأخير منخفض، ومن ثمَّ بأداء مرتفع للتطبيق، لكنه يتطلب حجماً كبيراً من المعلومات المضافة overhead لضمان بناء البنية التراكمية overlay، في حين يتطلب النمط الشبكي موازنة بين التأخير والمعلومات المضافة [15].

2.1. الهدف من البحث

يهدف هذا البحث إلى تصميم تطبيق ند لند واستخدامه من أجل مشاركة الملفات، وكذلك للبث الفيديوي بالزمن الحقيقي للاستفادة من مزايا هذه البنية، وتنفيذه برمجياً، وتحديد الزمن اللازم لاستقبال أنواع مختلفة من الملفات، بما في ذلك ملفات الفيديو، وكذلك تحديد زمن التأخير اللازم قبل بدء عرض الفيديو في حالة البث الفيديوي.

3.1. محددات البحث

على الرغم من المزايا الكثيرة التي تتمتع بها شبكات الند للند والتي أدت لاستخدامها في تطبيقات البث الفيديوي، إلا أنه يوجد العديد من التحديات الواجب أخذها بالحسبان عند بناء تطبيق بث فيديو يعتمد على شبكات الند للند.

يتمثل التحدي الأول في آلية اختيار العقد لتأمين عرض مجال كافٍ لتبادل المعلومات، وخاصة ملفات الفيديو ذات الحجم الكبير. ونظراً لاعتماد شبكة الند للند على عقد غير متجانسة من حيث عرض المجال، يصعب تأمين عرض المجال المطلوب لتبادل الملفات باستخدام عقدة واحدة، ويصبح لازماً الاتصال بأكثر من عقدة في آنٍ واحدٍ للحصول على كامل الملف.

وحيث أن الند قادر على الحصول على الملف من عدد من العقد، يتوجب تقسيم الملف إلى أجزاء في العقد المرسله ثم إعادة تجميعها وترتيبها في العقدة المستقبلية للحصول على كامل الملف.

إضافةً إلى ذلك، تشكل ديناميكية العقد تحدياً إضافياً عند بناء تطبيق يعتمد على بنية الند للند، حيث أن انضمام العقد ومغادرتها للشبكة يكون مستمراً وبدون إعلام مسبق. وتتطلب ديناميكية العقد هذه وجود آلية لإعادة بناء الشبكة بسرعة وكفاءة، ولاستعادة رزم packets المعطيات التي يمكن فقدانها.

4.1. الخلاصة

اعتمدت تطبيقات مشاركة الملفات والبث الفيديوي بادئ الأمر على بنية المخدم/زبون، ولكن مع ازدياد الطلب على هذه التطبيقات أصبح استخدام هذه البنية غير ملائم بسبب تصعدها المنخفض، الذي يعتمد على قدرات المخدم المسؤول عن إنجاز كافة عمليات الاتصال بالشبكة وموارده.

ظهرت فيما بعد بنية الند للند والتي تمتاز بتصعدها المرتفع وتكلفتها المنخفضة، حيث تتألف من عدة عقد تتشارك الموارد فيما بينها لإنجاز الأعمال المطلوبة. استخدمت شبكات الند للند، في البداية، من أجل مشاركة الملفات بين المستخدمين وظهرت العديد من التطبيقات في هذه المجال. وأدى النجاح الكبير الذي حققته هذه البنية إلى جذب اهتمام مطوري تطبيقات البث الفيديوي، حيث استفادوا من ميزات هذه البنية لتأمين إرسال الفيديو إلى عدد كبير من المستخدمين وبزمن تأخير محدود.

تقسم بنية الند للند إلى بنية صافية وبنية هجينة تحتوي على مخدم مركزي مسؤول عن تنظيم العقد والملفات ضمن الشبكة. تعاني البنية الصافية من صعوبة في الإدارة وعدم فعالية عملية البحث في حال الازدياد الكبير لعدد المستخدمين، أما في البنية الهجينة فتكون عملية الإدارة والبحث سريعة وفعالة. في التطبيق المقترح، قمنا باعتماد البنية الهجينة لتسهيل عملية الإدارة، كما قمنا بتقسيم الملف إلى أجزاء متساوية الحجم، وذلك لاستغلال الموارد بالشكل الأمثل.

شركات الند للند

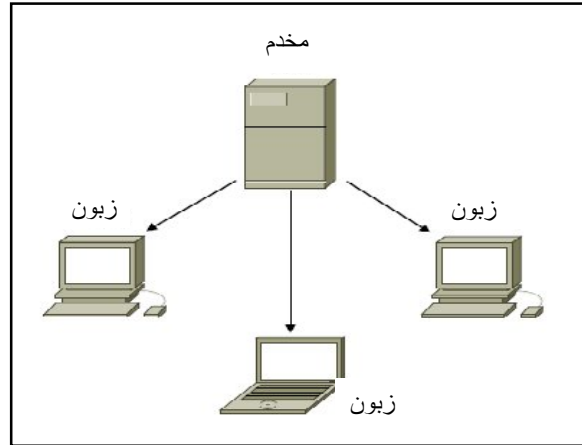
peer-to-peer networks

اعتمدت شبكات الاتصالات الحاسوبية خلال العقود السابقة بشكل أساسي على بنية المخدم/زبون، ولعل أشهرها النسيج العنكبوتي العالمي World Wide Web (WWW) لكن اعتماد بنية المخدم/زبون على وجود مخدم مركزي مسؤول عن كافة الأعمال، جعلها تعاني من نقص في التصعد والوثوقية reliability. وبهدف تجاوز هذه المشاكل ظهرت شبكات الند للند التي تعتمد على عدد من العقد لها نفس القدرات capabilities والمسؤوليات responsibilities، إذ يلعب كل منها دور المخدم والزبون بآنٍ واحد.

وأصبحت تطبيقات الند للند في العام 2006 تشغل حوالي 70% من التطبيقات على الانترنت [16] والتي تشمل مشاركة الملفات file sharing، والبث (الدفق) الفيديوي، وغيرها من الخدمات. يتضمن هذا الفصل بدايةً مقدمة عن شبكات المخدم/زبون، ثم شرحاً لبنية شبكات الند للند إضافةً إلى إجراء مقارنة بين هاتين البنيتين. كما يتضمن تصنيفاً لشبكات الند للند.

1.2. شبكة المخدم/زبون

تتألف الشبكة من مخدم، يتمتع بمواصفات عالية من حيث مساحة التخزين، وقدرات المعالجة، وعرض المجال، بالإضافة إلى عدة مستخدمين [17] كما هو مبين بالشكل (1-2).



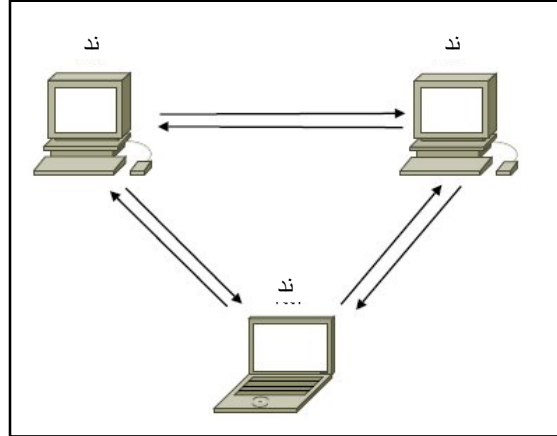
الشكل (1-2) مثال على بنية المخدم/زبون.

يمتاز هذا النوع من الشبكات بسهولة الصيانة maintenance حيث من الممكن تطوير المخدم، واستبداله وإصلاحه بسهولة، كما أن وجود البيانات في مكان واحد من الشبكة يُسهّل عملية التحديث update.

وتعاني هذه الشبكات بالمقابل من العديد من المشاكل منها وجود نقطة فشل واحدة single point of failure في المنظومة تتمثل بالمخدم، وكذلك زيادة زمن الاستجابة لطلبات المستخدمين بازدياد عددهم [1].

2.2. شبكة الند للند

تتألف شبكة الند للند من عدة عقد nodes لها نفس الصلاحيات والمسؤوليات، وتتشارك في مواردها سواء المادية (مثل الطابعة، ومساحة التخزين، وعرض مجال الشبكة وغيرها) أو المنطقية (كالخدمات) بهدف إنجاز مهام محددة مثل التشارك بالملفات، وتبادل الرسائل الفورية instant messaging والمعالجة الموزعة distributed computing [17، 18]. ويبين الشكل (2-2) مثالاً على بنية هذه الشبكة. يؤدي انضمام عقد جديدة إلى الشبكة إلى إضافة مواردها إلى الموارد المتاحة في الشبكة، ومن ثمّ يمتاز هذا النوع من الشبكات بتعدد كبير. كما أن وجود نفس البيانات في العديد من العقد يزيد من وثوقية المنظومة تجاه فشل بعض العقد أو مغادرتها الشبكة [12]. لكن هذه الميزات تزيد بالوقت نفسه من صعوبة إدارة الشبكة وتتسبب بانخفاض أمنها security. ويبين الجدول (2-1) مقارنة بين خصائص شبكة المخدم/زبون وشبكة الند للند.



الشكل (2-2) مثال على بنية شبكة الند للند.

الجدول (1-2) مقارنة بين شبكات الند للند وشبكات المخدم/زبون [19].

شبكة المخدم/زبون	شبكة الند للند	
	هجينة hybrid	صافية pure
غير متناظرة asymmetric: يقوم المخدم والزبائن بمهام مختلفة	غير متناظرة: يقوم المخدم والزبائن بمهام مختلفة	متناظرة symmetric: كل العقد تؤدي العمل نفسه
معرفة عامة global knowledge: يملك المخدم نظرة شاملة عن كامل الشبكة	معرفة عامة: يملك المخدم نظرة شاملة عن كامل الشبكة	معرفة محلية local knowledge: تملك العقدة معلومات عن مجموعة صغيرة فقط من العقد
مركزية: تكون الإدارة والاتصالات مركزة في المخدم	موزعة: تكون العمليات محلية ضمن العقد ويقتصر عمل المخدم على تخزين البيانات وتنظيم الشبكة	موزعة: تكون العمليات محلية ضمن العقد
وجود نقطة فشل واحدة: فشل المخدم يؤدي لتوقف كامل المنظومة	متانة جيدة: لا يؤدي فشل المخدم إلى توقف كامل المنظومة، وكذلك قد لا يؤثر فشل العديد من العقد على المنظومة بشكل واضح	متانة robustness عالية: فشل العديد من العقد قد لا يؤثر على المنظومة بشكل واضح
تصعد محدود: يعاني المخدم من التحميل الزائد overload بسرعة	تصعد عالي: بسبب توزيع العمل على المستخدمين	تصعد عالي: بسبب توزيع العمل على المستخدمين
تكلفة عالية للمخدم وعرض مجال محدود	تكلفة منخفضة وعرض مجال موزع بين المستخدمين	تكلفة منخفضة وعرض مجال موزع بين المستخدمين

3.2. مزايا تطبيقات الند للند

تُعدّ التكلفة المنخفضة، والفعالية efficiency، والمرونة flexibility من أهم مزايا تطبيقات الند للند [20].

1.3.2. التكلفة المنخفضة

تتجم التكلفة المنخفضة لشبكات الند للند بالمقارنة مع الشبكات المركزية (مخدم/زبون) عن إنقاص موارد الإدارة المركزية، وإنقاص موارد التخزين المطلوبة، وأمثلة موارد المعالجة.

أ. إنقاص موارد الإدارة المركزية

يكون المخدم المركزي في شبكات المخدم/زبون هو المسؤول عن جميع عمليات الإدارة والتنظيم بالشبكة وإعطاء الصلاحيات للمستخدمين، ويتطلب ذلك استخدام مخدم مركزي ذا موارد عالية ومن ثمّ تكلفة مرتفعة. أما بالنسبة لشبكات الند للند فتكون الإدارة موزعة، حيث تكون كل عقدة مسؤولة عن إدارة الملفات والموارد الخاصة بها وتضعها بخدمة الشبكة ككل.

ب. إنقاص موارد التخزين المطلوبة

في بنية المخدم/زبون يجري تخزين الملفات المراد مشاركتها ضمن المخدم، وهذا يتطلب استخدام مخدم بمساحة تخزين كبيرة، أما في حالة شبكة الند للند، فتخزن كل عقدة الملفات الخاصة بها بشكل محلي. ومن ثمّ تُعدّ مساحة التخزين للشبكة هي مجموع مساحات التخزين لكل عقدة.

ج. أمثلة موارد المعالجة

على عكس بنية المخدم/زبون، تكون المعالجة موزعة على جميع عقد بنية الند للند وليست محددة بالمخدم فقط.

2.3.2. الفعالية

لا يتطلب انضمام العقد إلى بنية الند للند أي عمليات تنصيب setup ابتدائية، وإنما تكنفي العقدة بالاتصال بأحد العقد ضمن البنية، ومن ثمّ تُضاف موارد هذه العقدة إلى موارد الشبكة وتبدأ عملية مشاركة الملفات.

3.3.2. المرونة

لا يتطلب إضافة وظائف إضافية لشبكة الند للند، أي عمليات إضافية وإنما يكفي تعديل العقدة المطلوبة وتطويرها، على عكس الحال في بنية المخدم/ زبون، حيث يتطلب ذلك تعديل المخدم المركزي من قبل إختصاصيين.

4.2. تحديات تطبيقات الند للند

هناك مجموعة من التحديات التي يجب أخذها بعين الحسبان عند بناء تطبيق ند للند ومنها [21].

1.4.2. مشاركة بيئة العمل

قد يؤدي اتصال عدة عقدة تعمل ضمن بيئات عمل مختلفة وغير متوافقة إلى مشاكل في أداء الشبكة.

2.4.2. ديناميكية الشبكة

تعاني شبكات الند للند من انضمام العقد ومغادرتها للشبكة بشكل مستمر وبدون إعلام مسبق، مما يقلل من وثوقية الشبكة.

3.4.2. ديناميكية خصائص العقد

بما أن خصائص العقد ضمن الشبكة قد تتغير بشكل ديناميكي، فقد يؤدي ذلك إلى التأثير على أداء التطبيق.

4.4.2. عدم تجانس الشبكة

وهو ينتج عن عدم تجانس العقد من حيث عرض المجال، أو موارد المعالجة أو مساحة التخزين المخصصة لكل عقدة.

5.2. تصنيف شبكات الند للند

يمكن أن تصنف شبكات الند للند تبعاً لاستخداماتها إلى [14]:

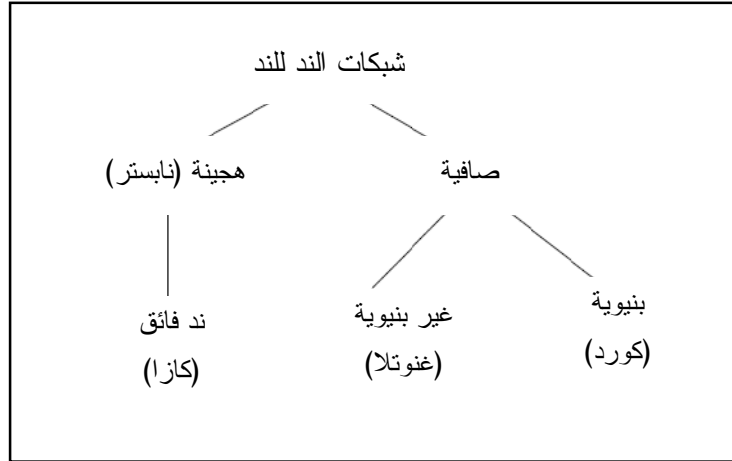
- مشاركة الملفات
- اتصالات هاتفية
- بث الوسائط المتعددة (صوت، فيديو)

كما يمكن أن تصنف تبعاً لدرجة مركزيتها degree of centralized إلى:

▪ شبكة صافية pure

▪ شبكة هجينة hybrid

ويبين الشكل (2-3) مخططاً لطرق تصنيف شبكات الند للند مع بعض الأمثلة.



الشكل (2-3) تصنيف شبكات الند للند [22].

1.5.2. الشبكة الصافية

نقول عن شبكة أنها صافية إذا تكونت من عناصر يعمل كل منها كمخدم وزبون بنفس الوقت من دون وجود مخدم مركزي مسؤول عن إدارة الشبكة وتنظيمها. وتمتاز هذه الشبكة بالوثوقية وانخفاض التكلفة بسبب عدم الحاجة إلى تجهيزات بمواصفات مرتفعة [23]. تُصنف هذه الشبكة اعتماداً على آلية اتصال العقد مع بعضها البعض في فئتين هما:

أ. الشبكة البنيفية structured: ويجري فيها استخدام بروتوكول من أجل تنظيم تخزين

البيانات ضمن الشبكة والوصول إليها بطريقة فعالة [1].

ب. الشبكة غير البنيفية unstructured: ويجري فيها بناء الشبكة التي تصل بين العقد

بطريقة عشوائية [1].

2.5.2. الشبكة الهجينة

تشتمل على مخدم مركزي مسؤول عن بناء الشبكة والتحكم بها. أما تخزين البيانات فيجري بإحدى طريقتين [6]:

أ. فهرسة مركزية centralized indexing: ونقوم على وجود قاعدة معطيات في المخدم تشتمل على أماكن تواجد البيانات ضمن العقد، حيث يقوم المخدم بتوجيه العقد الجديدة مباشرة إلى العقد التي تملك البيانات المطلوبة.

ب. فهرسة موزعة decentralized indexing: ويكون المخدم فيها مسؤولاً عن استقبال العقد الجديدة وإعطائها معلومات عن العقد الموجودة في الشبكة من أجل الاتصال بها والاستفسار عن البيانات المطلوبة.

6.2. التشارك في الموارد

تُعدّ مشاركة الموارد resource sharing من أهم مزايا شبكات الند للند، والتي أدت إلى انتشارها الواسع. فعند انضمام عقدة جديد إلى الشبكة، تضاف مواردها الفيزيائية، مثل مساحة التخزين وعرض المجال للشبكة، والمنطقية مثل الخدمات services و المعلومات، إلى موارد الشبكة [1].

1.6.2. عرض المجال

يُعدّ توفير عرض المجال من أهم المزايا التي تقدمها بنية شبكات الند للند، والذي يتحقق بواسطة تشارك العقد بعرض المجال الخاص بها من أجل ضمان توزيع البيانات ضمن الشبكة [6].

2.6.2. موارد المعالجة

تُعدّ عمليات المعالجة مورداً آخر يجري مشاركته في شبكات الند للند، حيث تقوم كل عقدة بتنفيذ مجموعة من التعليمات instructions أو بعض عمليات معالجة manipulated البيانات المطلوبة [6].

3.6.2. التخزين

تستخدم شبكات الند للند لبناء منظومة تخزين كبيرة، نتيجة تشارك جميع العقد بمساحة التخزين المتاحة ضمنها. فمثلاً من أجل شبكة تضم 100 عقدة، تقدم كل منها مساحة تخزين قدرها 1 غيغابايت (1Gbps)، تكون مساحة التخزين الإجمالية للمنظومة 100 غيغابايت [6].

7.2. الخلاصة

عرضنا في هذا الفصل بنية شبكات المخدم/زبون التي تتألف من مخدم مركزي مسؤول عن تخزين كامل المعلومات المراد مشاركتها، وعن تلبية جميع طلبات المستخدمين. وقد أشرنا إلى مزايا وجود مخدم مركزي في الشبكة من حيث سهولة الصيانة والتطوير، لكن المخدم المركزي يشكل نقطة فشل واحدة في المنظومة. كما تناولنا بنية شبكات الند للند التي تكون فيها جميع العقد متساوية بالقدرات والمسؤوليات والتي يلعب كل منها دور المخدم والزبون بنفس الوقت، وبيننا مزاياها من حيث التصعد الكبير والمتانة تجاه فشل أو مغادرة بعض العقد للشبكة.

كما قمنا بدراسة تصنيف شبكات الند للند، والتي تصنف تبعاً لدرجة مركزيتها، في فئتين: شبكة هجينة تشتمل على مخدم مركزي مسؤول عن بناء الشبكة والتحكم بها، وشبكة صافية تكون فيها جميع العقد متساوية بالقدرات والمسؤوليات، والتي تقسم بدورها إلى شبكات بنوية تستخدم بروتوكول محدد لتنظيم البيانات ضمن الشبكة، وغير بنوية يجري فيها بناء الشبكة بطريقة عشوائية.

شركات الند للند من أجل مشاركة الملفات

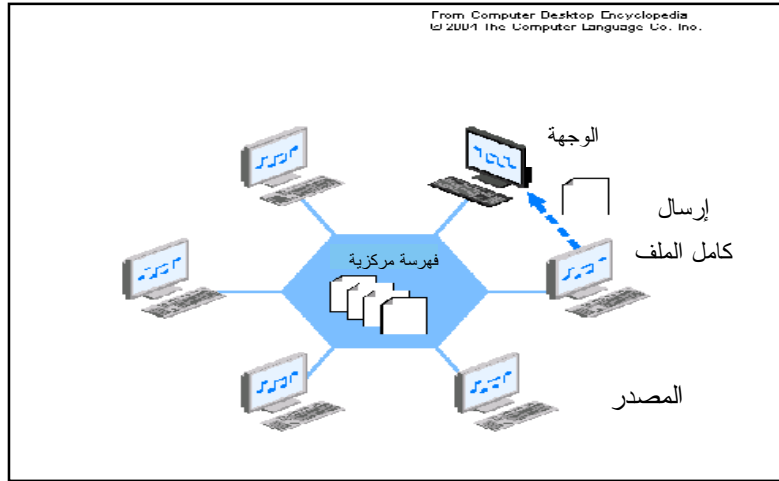
P2P networks for file sharing

تُعدّ تطبيقات مشاركة الملفات من أكثر تطبيقات الند للند شيوعاً. وتعتمد على تبادل الأنواع المختلفة للملفات، مثل الملفات النصية، والصوتية، والصور، والفيديو بين مستخدمي التطبيق. تسمى تطبيقات مشاركة الملفات بالفتح بعد الإتيان بالملف play after download، وهذا يعني ضرورة استقبال كامل الملف قبل فتحه أو تعديله. يتضمن الفصل بعض تطبيقات الند للند من أجل مشاركة الملفات.

1.3. بعض تطبيقات الند للند الشهيرة

1.1.3. التطبيق نابستر

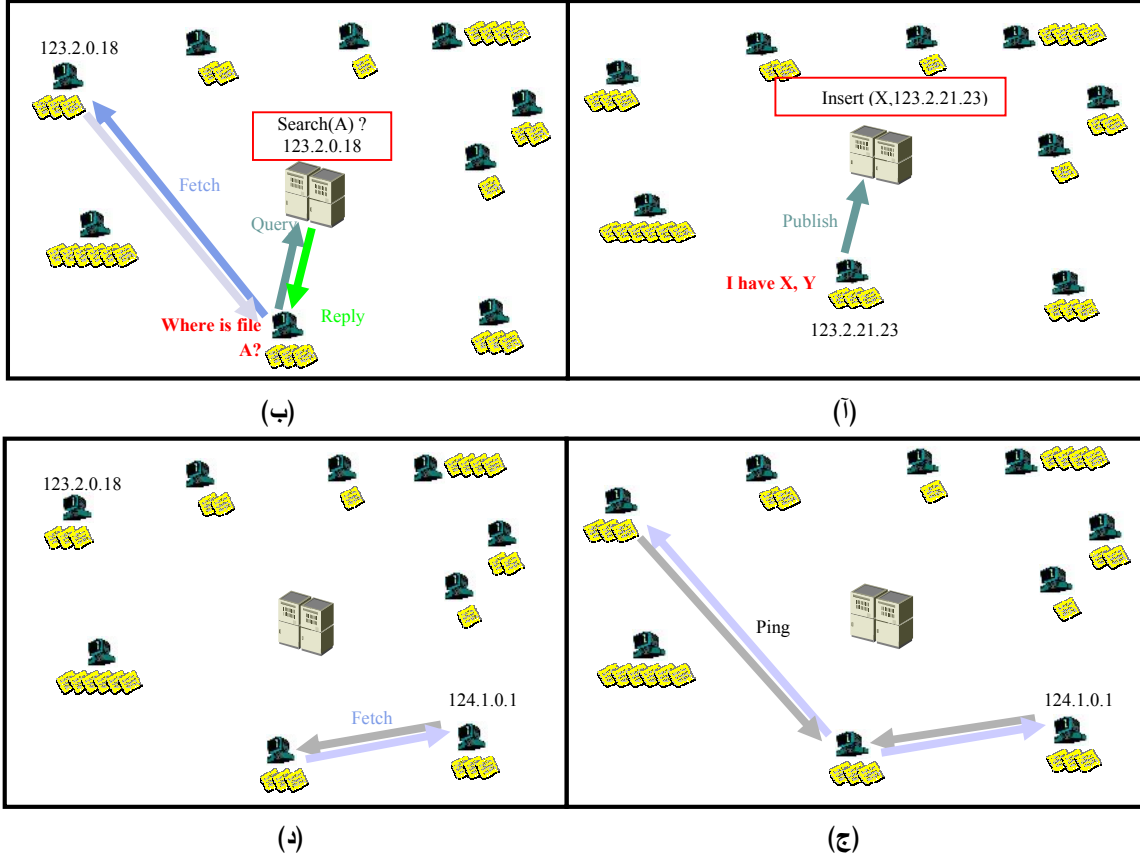
التطبيق نابستر تطبيق خاص بمشاركة ملفات الموسيقى من النوع MP3 عبر الإنترنت، ويُعد مثلاً على منظومة ند لند هجين، حيث يملك دليل مركزي centralized directory ينظم أسماء الملفات والأجهزة التي تضمها الشبكة. أما عملية تبادل الملفات فتتم بشكل مباشر بين عقدتين [18]. ويبين الشكل (3-1) بنية هذا التطبيق.



الشكل (3-1) بنية التطبيق نابستر [24].

أ. مبدأ عمله

عند انضمام مستخدم جديد للشبكة فإنه يقوم بالاتصال بالمخدم وإعلامه عن الملفات التي يرغب بمشاركتها، كما يمكن للمستخدم الاستفسار عن ملف محدد، ويرسل له المخدم لائحة بعناوين العقد التي تملك هذا الملف، يقوم المستخدم بعدها باختيار العقدة الأفضل والاتصال بها للحصول على الملف المطلوب [18]. ويبين الشكل (3-2) آلية عمل نابستر.



الشكل (2-3) آلية عمل التطبيق نابستر. (أ) مرحلة النشر publish، (ب) مرحلة البحث search، (ج) مرحلة الأخذ ping، (د) مرحلة الجلب fetch [24].

ب. محاسن التطبيق و مساوئه

يؤمن وجود الملفات في المخدم سرعة عملية البحث كما يزيد من فاعليتها، لكن وجودها في مخدم وحيد يشكل نقطة فشل في المنظومة ويؤدي إلى انخفاض في التصعد [18].

2.1.3. التطبيق غنوتلا

يُعدّ التطبيق غنوتلا Gnutella، وهو تطبيق لمشاركة الملفات، مثلاً على شبكات الند للند الصافية وغير البنيوية، فهو لا يتطلب وجود مخدم مركزي لفهرسة الملفات، إذ يمكن للمستخدم الحصول على مواضع تواجد الملفات وتحميلها من العقد الموجودة في الشبكة مباشرة [23]. ويعتمد هذا التطبيق على بروتوكول بسيط، يتألف من خمسة أنماط أساسية للرسائل والمبينة في الجدول (1-3).

الجدول (1-3) أنماط الرسائل في بروتوكول غنوتلا [18].

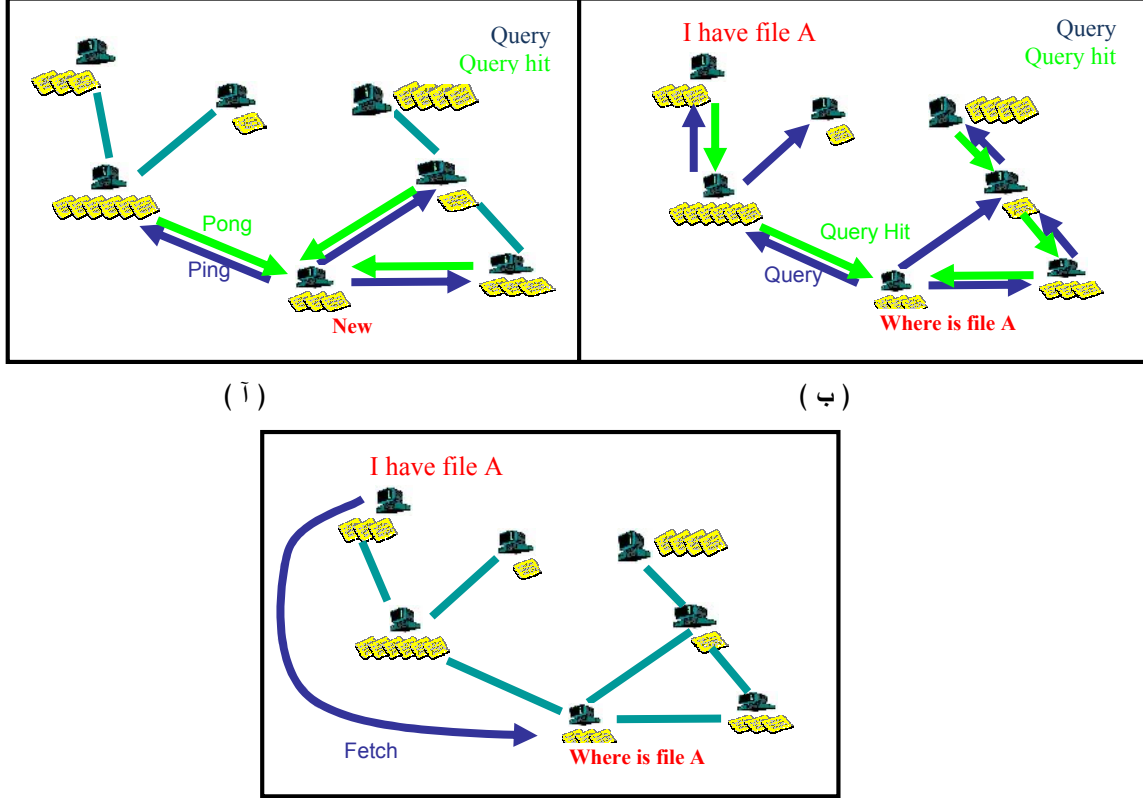
نمط الرسالة	توصيفها	المعلومات المتضمنة بها
أخذ Ping	إعلان عن العقدة واستكشاف العقد المجاورة لها	—
ردّ Pong	استجابة للأخذ	عنوان ورقم المنفذ port للعقدة المستجيبة بالإضافة لعدد الملفات المشاركة وعرض المجال الكلي لها
استفسار Query	طلب للبحث عن ملف ما	عرض المجال الأدنى للعقد المستجيبة بالإضافة لمعيار البحث
إصابة الاستفسار Query Hit	إجابة من العقد التي تملك الملف المطلوب	عنوان ورقم المنفذ للعقدة المستجيبة بالإضافة لنتائج البحث وعددها
دفع Push	طلب الإتيان بالملف من عقد تقع وراء جدار النار (الحماية) firewall	معين العقدة identifier، دليل الملف المطلوب، عنوان العقدة ورقم المنفذ المستخدم لإرسال الملف

أ. مبدأ عمله

عند انضمام مستخدم جديد للمنظومة فإنه يقوم بالاتصال بعقد محددة well-known، مثل gnutellhosts.com و router_limewire.com، ويقوم بعد ذلك بإرسال رسائل أخذ ping لاستكشاف العقد الموجودة في الشبكة، والتي تقوم بدورها بالرد عليه برسائل ردّ pong تضم معلومات عنها وعن الملفات التي تريد مشاركتها [25]. يرسل المستخدم بعدها رسالة استفسار query باسم الملف المطلوب إلى العقد المجاورة له، والتي تقوم بالبحث ضمن قائمة الملفات الخاصة بها. وهنا نميز حالتين: الأولى في حال وجود الملف ضمن العقدة، تقوم العقدة بإرسال رسالة إصابة الاستفسار (التي تبين وجود الملف في هذه العقدة) بنفس مسار رسالة الاستفسار إلى العقدة الطالبة ويجري بعد ذلك الاتصال بين العقدتين وتحميل الملف.

أما في حال عدم وجود الملف ضمن العقدة فإنها تقوم بإعادة توجيه الاستفسار إلى العقد المجاورة لها وذلك بعد إنقاص زمن حياة (TTL) Time To Live الاستفسار بمقدار 1 [18].

وتستمر عملية البحث والتي تسمى أيضاً تعويم flooding الاستفسار حتى الوصول إلى الملف المطلوب أو الوصول إلى قيمة زمن حياة مساوية للصفر [26]. ويبين الشكل (3-3) آلية عمل هذا التطبيق.



الشكل (3-3) مراحل عمل التطبيق غنوتلا، (أ) مرحلة استكشاف الشبكة، (ب) مرحلة البحث، (ج) مرحلة الجلب للملف [27].

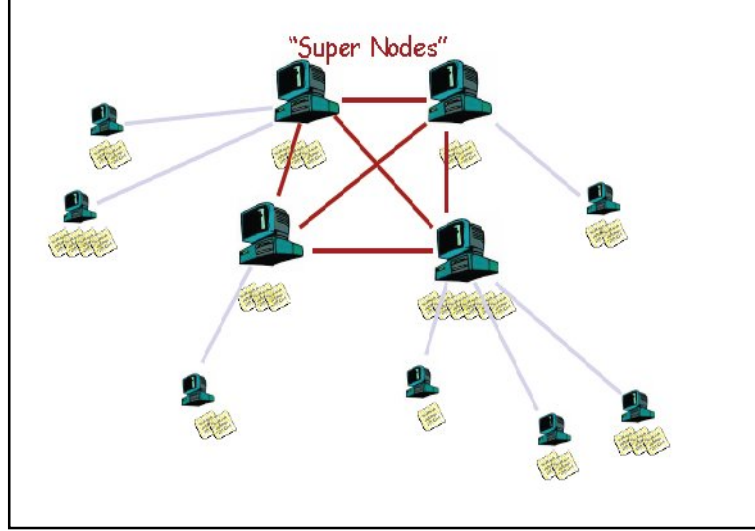
ب. محاسن التطبيق ومساوئه

يمتاز التطبيق غنوتلا بمناعة ضد الفشل fault-tolerant نتيجة وجود البيانات في العديد من العقد مما يزيد من وثوقية المنظومة ومتانتها، كما أن تكلفة البحث تكون موزعة على عدة عقد. في حين تعاني المنظومة من وجود تحميل زائد على الشبكة نتيجة تعويم الرسائل بين جميع العقد، وكذلك زيادة حجم المعالجة المتمثل في بحث كل عقدة عن الملفات ضمن قاعدة معطياتها database المحلية [18].

3.1.3. التطبيق كازا

يُعدّ التطبيق كازا KaZaA حالة خاصة من الشبكات الهجينة، فهو لا يملك مخدم مركزي لكن العقد ضمنه ليس لها ذات المسؤوليات. إذ تتألف المنظومة من نوعين من العقد هما العقد العادية ordinary والعقد الفائقة super، التي تمتاز عن الأولى بقدرات ومسؤوليات أكبر، كما هو مبين في

الشكل (3-4). ويجري اختيار العقد الفائقة اعتماداً على عرض المجال، وقدرات المعالجة، وقابلية الاتصال connectivity [28].



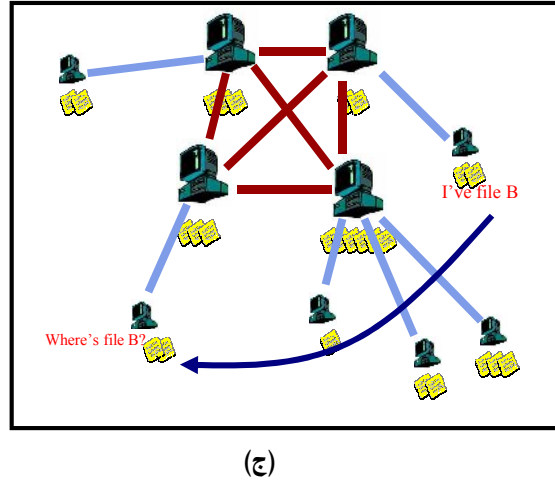
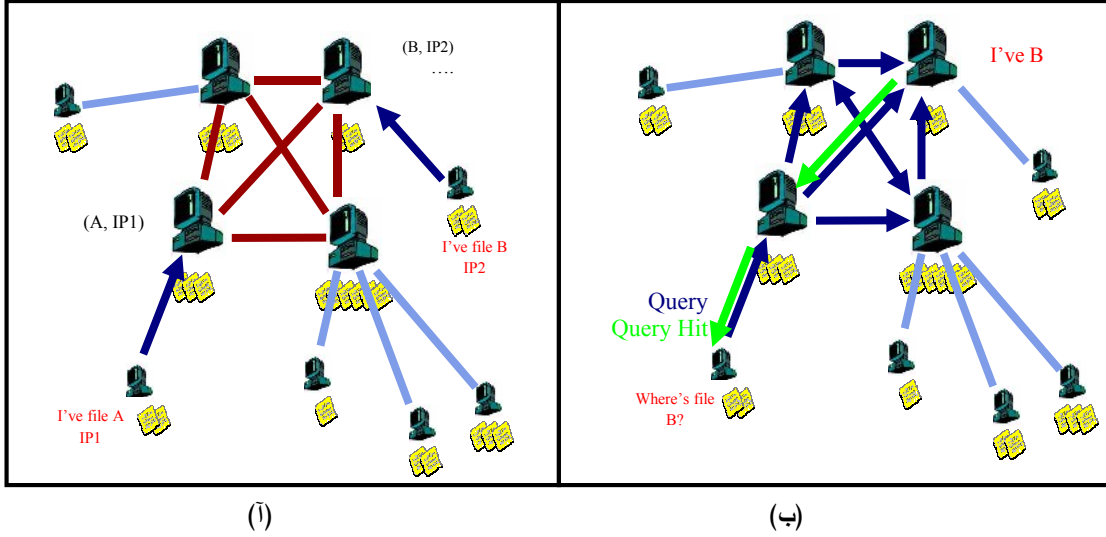
الشكل (3-4) بنية التطبيق كازا [27].

أ. مبدأ عمله

تتصل العقد الفائقة مع بعضها البعض مشكلة شبكة تراكيبية، بينما تتصل كل عقدة عادية بعقدة فائقة واحدة فقط. وفي حال انضمام مستخدم جديد للتطبيق، فإنه يرسل إلى أقرب عقدة فائقة لائحة بالملفات التي يرغب بمشاركتها، فتقوم العقدة الفائقة بإضافة هذه المعلومات إلى قاعدة بياناتها المتضمنة على معيّن الملف، ومعطيات مترفعة metadata للملف، وعناوين العقد العادية التي تضم هذا الملف.

وعند الحاجة إلى الاستفسار عن ملف ما، يوجه المستخدم الاستفسار إلى العقدة الفائقة التابع لها، والتي تقوم بدورها بالبحث عن الملف ضمن قاعدة بياناتها، وفي حال وجوده تقوم بإرسال عناوين العقد التي تملك الملف ومعلومات الملف للعقدة الطالبة. أما في حال عدم وجوده فإنها تقوم بتعويض الاستفسار بين العقد الفائقة للوصول إلى الملف المطلوب.

عند استقبال المستخدم لعناوين العقد المرغوبة فإنه يقوم بالاتصال بواحدة أو أكثر من هذه العقد للحصول على الملف المطلوب [29]. ويبين الشكل (3-5) آلية عمل التطبيق كازا.



الشكل (3-5) آلية عمل التطبيق كازا. (أ) مرحلة النشر، (ب) مرحلة البحث عن الملف، (ج) مرحلة جلب الملف [27].

ب. محاسن التطبيق ومساوئه

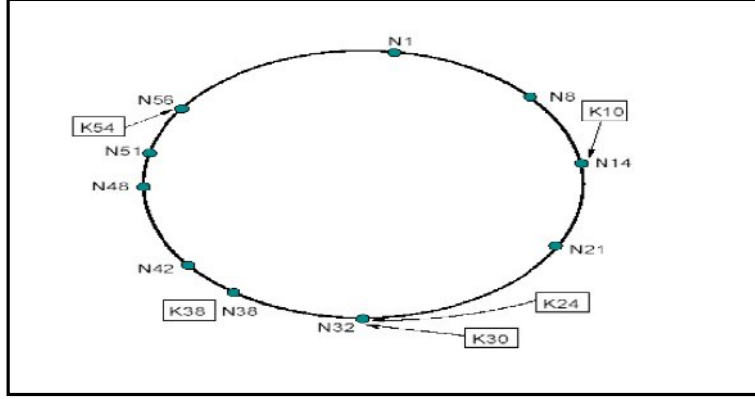
يأخذ التطبيق بالحسبان عدم تجانس heterogeneity العقد من حيث عرض المجال، وقدرات المعالجة، والمتاحة availability. لكنه يعاني من وجود تحميل زائد على بعض العقد (الفائقة)، وعدم إمكانية التحكم بزمان البحث ومجاليه.

4.1.3. التطبيق كورد

يُصنف التطبيق كورد Chord كأحد تطبيقات الند للند البنيوية الموزعة، ويستخدم أيضاً في مشاركة الملفات. يعتمد مبدأ عمله على إعطاء معين بطول m-bit لكل عقدة من خلال تشفير

عنوان بروتوكول الانترنت (IP) الخاص بها بتقنية التشفير السري ⁽¹⁾ SHA، وكذلك معيّن للمفتاح من خلال تشفيره بنفس التقنية [30].

وتكون المعيّّنات موزعة على دائرة بطول 2^m تدعى حلقة كورد Chord ring [31]، أما المفاتيح فتكون موزعة على العقد بحيث يكون المعيّّن للعقدة مساوياً أو أكبر من المعيّّن للمفتاح ، كما هو مبين في الشكل (3-6).



الشكل (3-6) توزيع العقد والمفاتيح على حلقة كورد [26].

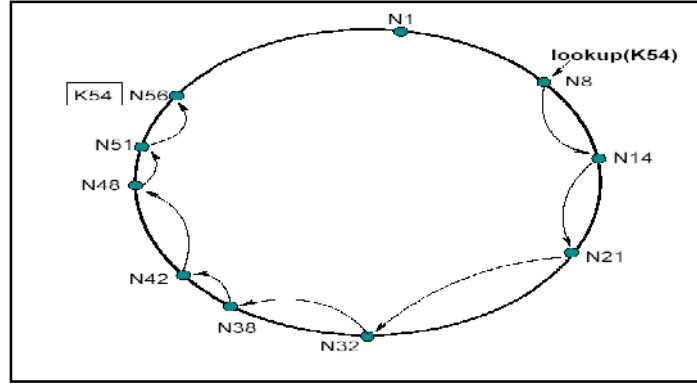
أ. طرق توزيع المفاتيح

هناك طريقتين لتوزيع المفاتيح key location هما الطريقة البسيطة، والطريقة المتصعدة .scalable

الطريقة البسيطة لتوزيع المفاتيح:

في هذه الطريقة تملك العقدة معلومات عن العقدة التي تليها مباشرة بالمعّين، وعند الاستفسار عن مفتاح معين فإنه يجري البحث ضمن العقدة أولاً، وفي حال عدم وجود المفتاح ضمنها يجري توجيه الاستفسار إلى العقدة اللاحقة. وبذلك تزداد تكلفة البحث بشكل خطي مع زيادة عدد العقد في المنظومة [26]. يبين الشكل (3-7) آلية البحث عن المفتاح بالطريقة البسيطة.

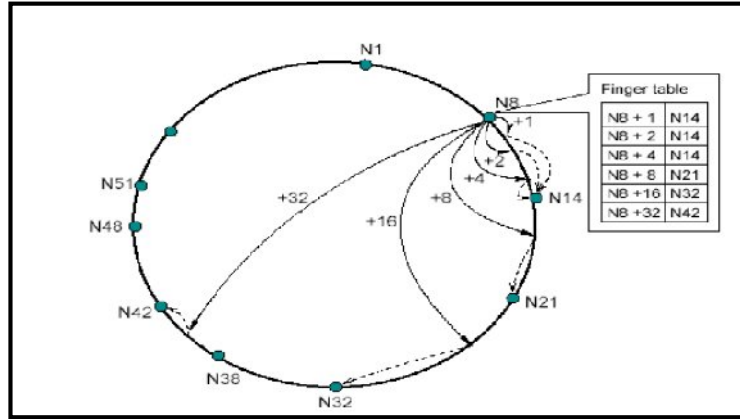
¹ خوارزمية التشفير السري Secure Hash Algorithm تستخدم من أجل تمثيل الرسائل أو ملفات المعطيات. حيث تنتج خوارزمية التشفير السري من أجل رسالة دخل لا يتجاوز طولها 2^{64} بت، خرج مؤلف من 160 بت، وهي تدعى أرقام الرسالة message digest. تستخدم هذه الخوارزمية من أجل إيجاد الرسالة الممثلة بأرقام رسالة معطاة، أو من أجل المقارنة بين رسالتين تنتجان نفس أرقام الرسالة.



الشكل (7-3) البحث بالطريقة البسيطة ضمن حلقة كورد [26].

الطريقة المتصعدة لتوضع المفاتيح:

تؤمن طريقة فعالة للبحث حيث تضم كل عقدة جدول توجيه يضم معلومات عن m عقدة. وعند الاستفسار عن مفتاح معين فإنه يجري البحث ضمن جدول التوجيه، وفي حال عدم الوصول إلى معلومات عن العقدة المطلوبة فإنه يجري توجيه الاستفسار إلى العقدة الموجودة في نهاية جدول الدلالة $finger\ table$ ، وهكذا حتى الوصول إلى الملف المطلوب [26]. يوضح الشكل (8-3) آلية البحث باستخدام الطريقة المتصعدة.



الشكل (8-3) البحث باستخدام الطريقة المتصعدة [26].

ب. خصائص التطبيق

- التوزيع: تملك جميع العقد نفس الأهمية، لذا تمتاز المنظومة بالوثوقية لعدم وجود نقطة فشل واحدة فيها.
- المتاحية: إذ يعمل البروتوكول بشكل جيد حتى مع التغيرات المستمرة في المنظومة.
- التصعد: في حالة الطريقة المتصعدة، تزداد تكلفة البحث بشكل لوغاريتمي بدلالة عدد العقد في المنظومة وليس بشكل خطي، ومن ثم يناسب هذا التطبيق المنظومات الكبيرة.

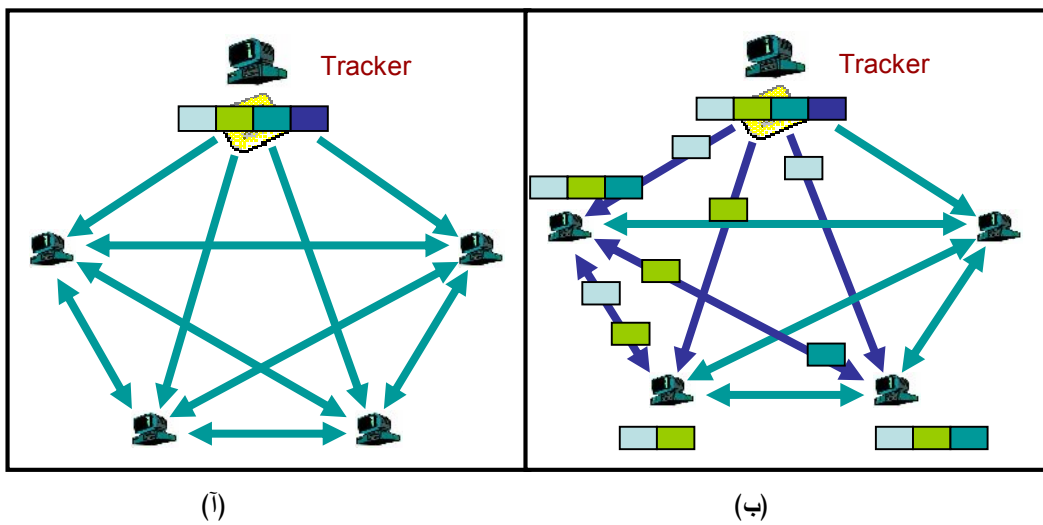
- موازنة الحمل load balance: يجعل استخدام عملية التشفير المفاتيح موزعة بشكل متساوٍ على العقد.
- تسمية مرنة flexible naming: لا يوجد أي محددات على بنية المفاتيح مما يتيح مرونة في تسمية الملفات من قبل المستخدم [26].

5.1.3. التطبيق بت تورنت

يُعدّ التطبيق بت تورنت BitTorrent من أشهر تطبيقات الند للند المستخدمة في تحميل الملفات بسرعة وكفاءة، حيث يعتمد تقسيم الملف إلى أجزاء متساوية الحجم تدعى كتل وذلك لتسهيل عمليات الإرسال وتسريعها. وتحتوي المنظومة على عنصر مركزي يدعى متعقب tracker وهو مسؤول عن تنظيم لائحة العقد التي تقوم بتحميل الملف عند تلك المرحلة وتحديث البيانات حول أجزاء الملف التي يقوم كل ند بتنزيلها أو تحميلها upload إلى العقد الأخرى [30، 32].

أ. مبدأ عمله

في البداية يقوم المستخدم بالبحث عن عنوان المتعقب للملف المطلوب، باستخدام أحد محركات البحث (مثل غوغل google)، ثم الاتصال به للحصول على لائحة بالعقد الواجب عليه الاتصال بها لتحميل الملف. ثم تبدأ مرحلة تسمى بالحشد swarming، حيث يقوم كل ند بتحميل أجزاء الملف التي يحتاجها من العقد التي تمتلكها، وكذلك إرسال الأجزاء التي يمتلكها إلى العقد التي تحتاجها [30]. ويبيّن الشكل (3-9) بنية التطبيق وآلية عمله.



الشكل (3-9) التطبيق بت تورنت (أ) بنية التطبيق، (ب) آلية عمل التطبيق [27].

ب. محاسن التطبيق ومساوئه

هذا التطبيق يمتاز بسرعة وكفاءة في عملية التنزيل نتيجة تقسيم الملف إلى أجزاء والتشارك في موارد العقد الموجودة ضمن المنظومة، لكنه يعاني من وجود نقطة فشل واحدة في المنظومة تتمثل بالمتعقب [30].

2.3. الخلاصة

استعرضنا في هذا الفصل أهم تطبيقات الند للند المستخدمة من أجل مشاركة الملفات مثل التطبيق نابستر المستخدم للتشارك في ملفات الموسيقى، والتطبيق غنوتلا الذي يُعد مثلاً على شبكة صافية وغير بنىوية لتشارك الملفات، والتطبيق كازا الذي يُعد حالة خاصة من الشبكات الهجينة التي تضم نوعين من العقد فائقة وعادية، والتطبيق كورد وهو أحد تطبيقات الند للند الموزعة والمستخدم لتشارك الملفات، وأخيراً التطبيق بت تورنت الذي يُعدّ من أشهر تطبيقات الند للند المستخدمة لتحميل الملفات بسرعة وكفاءة.

شركات الند للند من أجل البث الفيديوي

P2P networks for video streaming

تتجه تطبيقات البث الفيديوي في الوقت الراهن إلى شبكات الند للند، نظراً للعديد من الميزات التي تتمتع بها مثل التصعد الكبير والتكلفة المنخفضة. تعتمد تطبيقات الند للند على مساهمة كل عقدة في إرسال الفيديو إلى عقد أخرى مما يزيد من كفاءة المنظومة وقوتها مع ازدياد عدد المستخدمين. يستعرض هذا الفصل ميزات استخدام شبكات الند للند من أجل بث الفيديو وأهم التطبيقات المعتمدة على هذه البنية.

1.4. تقنيات التعامل مع الفيديو

هناك ثلاث تقنيات أساسية للتعامل مع الفيديو، وهي البث التلفزيوني broadcasting، والبث الفيديوي video streaming، وتبادل الفيديو video transferring.

البث التلفزيوني: يجري إرسال الفيديو إلى عدد كبير من المستخدمين عبر الأقمار الصناعية، أو الكابلات، أو خطوط النقل الأرضية Terrestrial.

البث الفيديوي: يجري إرسال الفيديو إلى عدد من المستخدمين عن طريق شبكة الانترنت، وفي هذه الحال يستطيع المستخدم مشاهدة الفيديو بالزمن الحقيقي ولا يحتاج إلى الانتظار حتى استقبال الفيديو كاملاً.

تبادل الفيديو: في هذه الحالة يتم إرسال واستقبال ملفات الفيديو عبر شبكة الانترنت، ويحتاج المستخدم إلى استقبال كامل الفيديو قبل ان يتم فتحه أو استخدامه.

2.4. تحديات شبكات الند للند من أجل البث الفيديوي

- هناك العديد من التحديات الواجب أخذها بالحسبان عند تصميم تطبيقات بث فيديوي اعتماداً على شبكات الند للند، ومن هذه التحديات [33، 34]:
- تتطلب خدمات الفيديو عرض مجال كبير لا يمكن تأمينه من قبل ند واحد، ومن ثمّ يحتاج الند المستقبل للاتصال بعدد من المرسلين للحصول على كامل الفيديو. تُعدّ عملية الاختيار أحد التحديات عند تصميم تطبيق بث فيديوي اعتماداً على شبكات الند للند.
 - تجعل الحاجة لبث الفيديو في الزمن الحقيقي من الضروري تأمين استقبال عينات الفيديو بدون تأخير يذكر، حيث يؤثر التأخير الكبير على جودة تطبيقات البث الفيديوي.

- تعتمد تطبيقات بث الفيديو على ترميز الفيديو قبل إرساله عبر شبكة الند للند إلى المستقبل. ويؤثر الاختيار المناسب لتقنية الترميز المعتمدة على كمية المعلومات المضافة، ومن ثم على جودة التطبيق.
- تُعدّ شبكات الند للند ذات تصعد عالي، فهي تتألف من عدد كبير من العقد غير المتجانسة heterogeneous، لذا يجب الأخذ بالحسبان، عند بناء تطبيق ند لند، عدم التجانس هذا وذلك لإيصال الفيديو إلى جميع المستخدمين بسرعة وكفاءة.
- تعاني شبكات الند للند من انضمام العقد ومغادرتها للشبكة بشكل مستمر وبدون إعلام notification مسبق، لذلك يجب الأخذ بالحسبان هذه الديناميكية عند تصميم تطبيق بث فيديوي يعتمد شبكة الند للند لضمان جودة عالية للفيديو المستقبل.

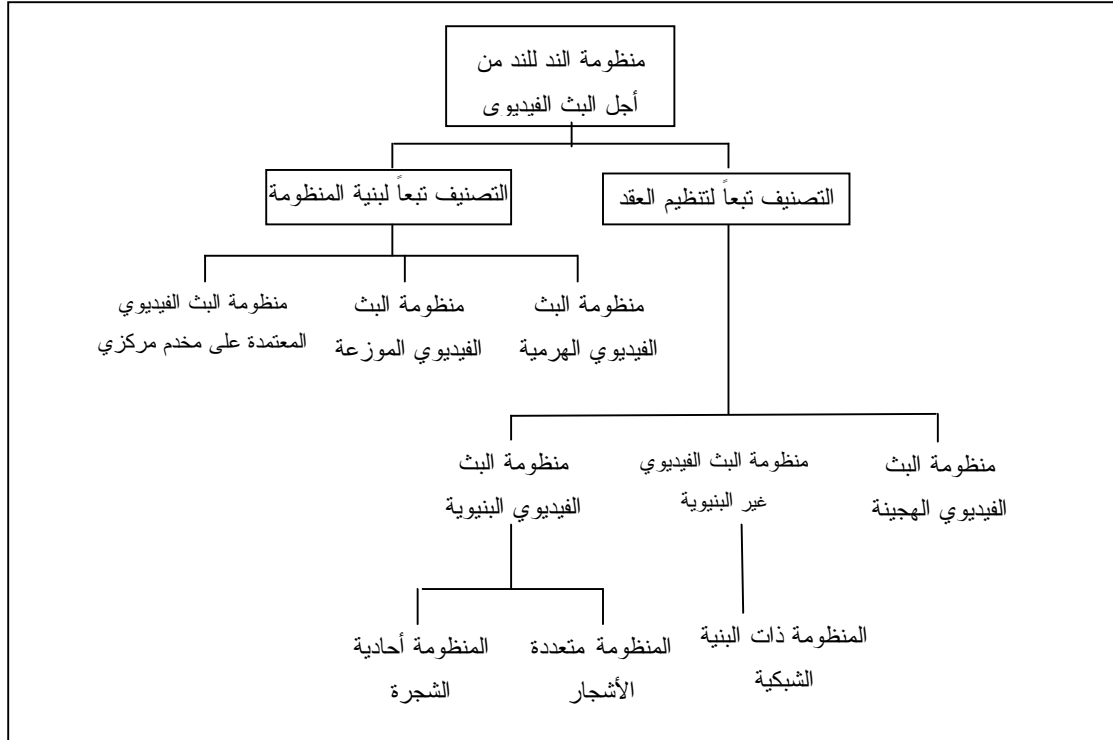
3.4. تصنيف تطبيقات الند للند من أجل بث الفيديو

تُصنف تطبيقات البث الفيديوي المعتمدة على شبكات الند للند تبعاً لـ [33]:

• توضع العقد ضمن الشبكة التراكمية

• بنية المنظومة

يبين الشكل (4-1) تصنيف تطبيقات الند للند من أجل البث الفيديوي.



الشكل (4-1) تصنيف تطبيقات الند للند من أجل بث الفيديو [33].

1.3.4. تصنيف تطبيقات الند للند لبث الفيديو اعتماداً على توضع العقد

يقسم هذا النوع من التطبيقات بدوره إلى منظومات بنويوية، وغير بنويوية، وهجينة [35].

1.1.3.4. منظومات البث الفيديوي البنويوية

أ. منظومات البث أحادية الشجرة

تتوضع العقد في هذه المنظومات ضمن بنية هرمية على شكل شجرة، حيث يتوضع الند الذي يمثل مصدر الفيديو عند جذر الشجرة بينما تتوزع بقية العقد عند الأطراف. وعند انضمام ند جديد للمنظومة يجري وضعه في المستوى المناسب من الشجرة. يمتاز هذا التطبيق بسهولة التنفيذ العملي، والحاجة لكمية قليلة من المعلومات المضافة من أجل تنظيم العقد. كما يمكن إنقاص التأخير الزمني والتحكم به بجعل عدد مستويات العقد في الشجرة محدود وأقل ما يمكن. تعاني المنظومة من بعض السيئات ومنها صعوبة إعادة بناء الشجرة بسرعة في حال ازدياد عدد العقد التي تتضمن أو تغادر المنظومة بصورة مستمرة، ويتسبب عدم تجانس العقد من حيث عرض مجال الوصلة الهابطة (الداخل إليها) downlink والوصلة الصاعدة (الخارج منها) uplink في ازدياد التأخير في بعض العقد. من أشهر التطبيقات العملية التي تعتمد على هذه البنية التطبيق سبريد إت SpreadIT ، والتطبيق بيركاست PeerCast، والتطبيق أوفر كاست Overcast ، إضافة لتطبيق نايس NICE والذي سنعرض له في الفقرة (1.4.4).

ب. منظومات البث متعددة الأشجار

تُعدّ منظومات البث أحادية الشجرة هشة تجاه ديناميكية العقد وعدم تجانسها. تعاملت منظومات البث متعددة الأشجار مع هاتين المشكلتين بواسطة تنظيم العقد في عدة أشجار. يحدد كل ند العدد المناسب من الأشجار التي سينضم إليها تبعاً لعرض المجال الذي يمتلكه، حيث يجب أن يكون عقدة وسيطة في شجرة واحدة على الأقل.

تعتمد المنظومة على تقسيم الفيديو إلى أجزاء sub-streams توزع بين عقد كل مستوي من مستويات الشجرة. تمتاز منظومات البث متعددة الأشجار بمتانة عالية حيث تتشارك كل العقد بمواردها مع الآخرين، إضافةً إلى إنقاص التأخير بالمقارنة مع المنظومات أحادية الشجرة. لكنها تعاني من تعقيد كبير في تنظيم العقد ضمن أشجار متعددة [33، 36]. يُعدّ كلاً من التطبيق سبلت ستريم SplitStream و التطبيق كوبنت CoopNet من أشهر تطبيقات البث المعتمدة على البنية متعددة الأشجار.

2.1.3.4. منظومات بث الفيديو غير البنيوية المعتمدة على شبكات الند للند

تتصل العقد ضمن المنظومة بشكل عشوائي بدون أي بنية محددة، حيث يتشارك أي ند الفيديو مع العديد من العقد تبعاً لعرض مجال الوصلة الهابطة والوصلة الصاعدة. يجري في المنظومات غير البنيوية تقسيم الفيديو إلى العديد من الأجزاء الصغيرة والتي تدعى الكتل، حيث يمكن للند استقبال هذه الكتل من عدة مصادر. وعلى عكس المنظومات البنيوية فإنه لا يوجد مسار محدد لتوزيع المحتويات حيث يمكن للمستخدم أن يطلب كتل الفيديو من أي ند آخر في حال فشل المصدر بالقيام بذلك. تُعد سهولة إصلاح المنظومة في حال مغادرة بعض العقد إضافة إلى زمن التأخير الصغير من أهم ميزات هذا النوع من المنظومات. أما السيئة الأساسية لهذه المنظومات فتتمثل بالحجم الكبير للمعلومات المضافة بسبب المراقبة المستمرة لكل من حالة الشبكة والعقد. ومن الأمثلة على هذا النوع من المنظومات التطبيق تشين ساو ChainSaw، والتطبيق برايم PRIME، والتطبيق ناردا Narada، والتطبيق دونت/كول ستريمينغ Donet/CoolStreaming.

3.1.3.4. منظومات البث الفيديوي الهجينة المعتمدة على شبكات الند للند

تمتلك هذه المنظومات خصائص كلاً من المنظومات البنيوية وغير البنيوية، ويُعد التطبيق بوليت Bullet مثالاً على هذا النوع من المنظومات، والذي يُستخدم من أجل إرسال ملفات كبيرة الحجم، وكذلك للبث الفيديوي لعدد كبير من المستخدمين. يعتمد تطبيق بوليت في عمله على تقسيم الفيديو إلى كتل صغيرة حيث يقوم الند باستقبال كتل الفيديو من أبائه في الشجرة في حال وجودها، أما في حال فقدان بعض الكتل فإنه يستقبلها من عقد أخرى [33].

2.3.4. تصنيف منظومات الند للند من أجل البث الفيديوي تبعاً لبنية المنظومة

تصنف منظومات البث تبعاً لبنية المنظومة إلى ثلاث مجموعات أساسية هي: البنية ذات المخدم المركزي central server based، والبنية الموزعة distributed based، والبنية الهرمية hierarchical based.

1.2.3.4. منظومات الند للند ذات المخدم المركزي من أجل بث الفيديو

تعتمد هذه المنظومات على وجود مخدم مركزي من أجل إدارة العقد وبناء الشجرة. وتعاني هذه المنظومات من وجود نقطة فشل واحدة في المنظومة تتمثل بالمخدم المركزي. ومن الأمثلة على هذا النوع تطبيق كوبنت CoopNet.

2.2.3.4. منظومات الند للند الموزعة من أجل البث الفيديوي

على عكس المنظومات ذات المخدم المركزي، تكون إدارة العقد وبناء الشجرة في هذه المنظومات موزعة. ومن الأمثلة عليها التطبيق سبريد إت ، والتطبيق بالس PALS، والتطبيق برومايس PROMISE، والتطبيق سبلت ستريم.

3.2.3.4. منظومات الند للند الهرمية من أجل البث الفيديوي

يجري تنظيم العقد في هذه المنظومات ضمن عدة مستويات هرمية مما يزيد من قدرة المنظومة على التصعد. يُعدّ تطبيق زيك زاك ZigZag أحد الأمثلة على هذا النوع من المنظومات [33].

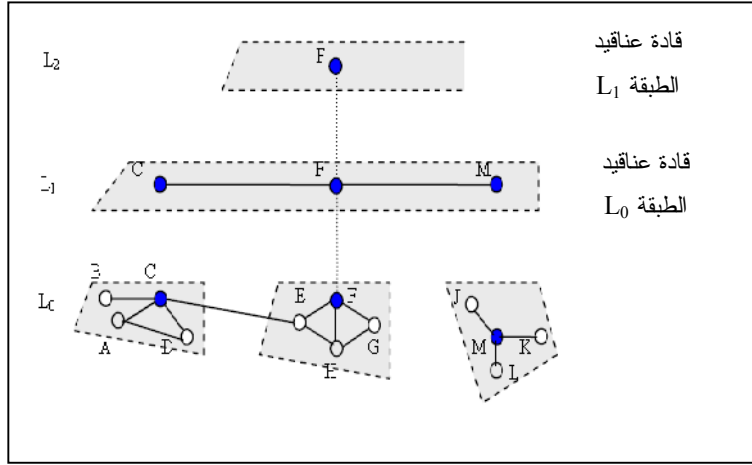
4.4. بعض تطبيقات الند للند المستخدمة من أجل بث الفيديو

1.4.4. التطبيق نايس

يستخدم التطبيق نايس تقنية هرمية لدعم عدد كبير من المستخدمين، ويُعدّ مثلاً على المنظومات أحادية الشجرة. وتستخدم في التطبيقات ذات عرض المجال المنخفض وتطبيقات البث بالزمن الحقيقي.

تعتمد بنية التطبيق نايس على تنظيم العقد المتقاربة من حيث التأخير الزمني latency ضمن عناقيد clusters يتراوح حجمها بين $(k, 3k-1)$ حيث k عدد ثابت يمثل حجم العنقود. وفي حال ازدياد حجم العنقود عن $3k-1$ فإنه يجري تقسيمه إلى عنقودين بحجم مساوي لـ $3k/2$ تقريباً، أما في حال نقصان حجمه عن k فإنه يجري دمج عقد من العنقود المجاور [37].

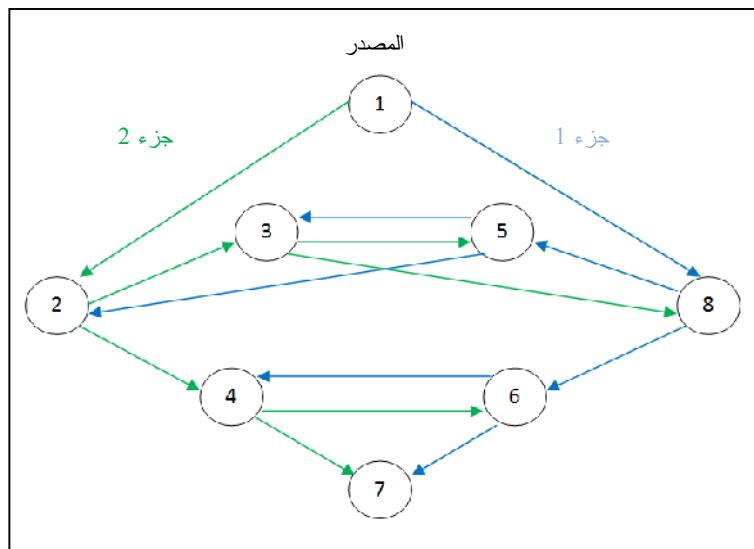
تكون جميع العقد منتمية إلى الطبقة الدنيا من البنية الهرمية L_0 . بينما تضم الطبقة اللاحقة L_1 قادة عناقيد الطبقة السابقة، حيث يجري اختيار قائد العنقود بحيث تكون المسافة بينه وبين بقية عقد العنقود أصغر ما يمكن. وتستمر هذه العملية حتى الوصول إلى قمة الهرم والتي تضم عنقود مؤلف من عقدة واحدة فقط. يكون عدد الطبقات مساوي لـ $\log_k N$ كحد أقصى حيث تمثل N العدد الكلي لعناصر التطبيق. يبيّن الشكل (4-2) بنية التطبيق نايس.



الشكل (2-4) بنية التطبيق نايس [38].

2.4.4. التطبيق سبلت ستريم

يُعدّ التطبيق سبلت ستريم من منظومات البث الموزعة والتي تعتمد على بنية متعددة الأشجار. ويعتمد على تقسيم الفيديو إلى أجزاء، يمرر كل منها من المرسل إلى المستقبل عبر شجرة مستقلة واحدة. يمثل كل ند عقدة وسيطة في شجرة واحدة فقط. ومن ثمّ فإن فشل أي ند يؤثر في جزء واحد فقط من الفيديو بحيث يمكن استرجاعه من بقية المصادر [33]. يوضح الشكل (3-4) المبدأ الأساسي لعمل التطبيق سبلت ستريم.



الشكل (3-4) المبدأ الأساسي لعمل التطبيق سبلت ستريم [12].

3.4.4. التطبيق ناردا

يُعدّ التطبيق ناردا مثالاً على المنظومات غير البنيوية والتي تسمى أيضاً المنظومات الشبكية، وهو ملائم لتطبيقات المؤتمرات عبر الإنترنت Internet conferencing. يجري بناء الشبكة التراكبية بواسطة إضافة الوصلات وإلغاءها بشكل دوري بين العقد تبعاً لرسائل تحكم متبادلة بينها. يقوم الند بمراقبة الوصلات وبناء جدول التوجيه routing table الخاص به اعتماداً على خوارزمية المسار الأقصر shortest widest path algorithm والتي تأخذ بالحسبان كلاً من عرض المجال والتأخير كمحددات أولي وثنائي على الترتيب. عندما يرغب الند بالانضمام للمنظومة فإنه يختار لائحة من العناصر الموجودة مسبقاً في نقطة التجميع rendezvous point، وبعدها يقوم باختيار مجموعة عشوائية من هذه العناصر للاتصال معها [39].

4.4.4. التطبيق بوليت

يُعدّ التطبيق بوليت من أشهر منظومات البث الهجينة التي تملك مواصفات كل من منظومات البث البنيوية التي تعتمد على البنية الشجرية، ومنظومات البث غير البنيوية التي تعتمد البنية الشبكية. يعتمد التطبيق بوليت في عمله على تقسيم الفيديو إلى كتل صغيرة، حيث يقوم الند باستقبال هذه الكتل من أبائه في الشجرة في حال تواجدهم، وإلا فإنه يحصل على الكتل المفقودة من أي عقدة أخرى. تتمتع هذا المنظومة بإمكانية اختيار أفضل ند حيث تقوم بتقييم العقد المرسلة له بشكل دوري وفي حال وجود عقدة جديدة أفضل فإنها تقوم بالانتقال إليها [33].

5.4.4. التطبيق كوبنت

يُستخدم التطبيق كوبنت في تطبيقات بث الفيديو والفيديو عند الطلب، وتُعدّ مثالاً على المنظومات المعتمدة على مخدم مركزي. يشتمل التطبيق على مخدم مركزي مسؤول عن تنظيم العقد ضمن بنية متعددة الأشجار. يستخدم التطبيق كوبنت تقنية الترميز المتعدد التوصيفات Multi Description Coding (MDC) من أجل ترميز الفيديو، حيث تعتمد هذه التقنية على ترميز الفيديو بعدة توصيفات مستقلة ترسل كل منها عبر شجرة واحدة. يستطيع المستقبل الحصول على كامل التوصيفات بالحالة المثالية، ومن ثمّ الحصول على كامل جودة الفيديو، أما في حال فقدان بعض هذه التوصيفات بسبب فشل بعض العقد، فإن ذلك يؤدي لانخفاض جودة الفيديو المستقبل [33، 37، 40].

6.4.4. التطبيق سبريد إت

يُعدّ التطبيق سبريد إت مثلاً على منظومات البث الموزعة، حيث تكون العقد هي المسؤولية عن تنظيم الشجرة. عند انضمام أي عقدة جديدة للتطبيق فإنها تبحث من أعلى الشجرة إلى أسفلها عن عقدة غير مشبعة unsaturated من حيث عرض المجال للاتصال بها. عند مغادرة عقدة للتطبيق فإنها ترسل لأبنائها رسالة للاتصال بالعقدة الأب لها، أما في حالات مغادرة العقد بشكل مفاجئ فيجري استخدام آلية لإعادة بناء الشجرة مما يتطلب فحصاً دورياً لحالة العقد.

يعاني التطبيق بشكل كبير من مغادرة العقد بشكل مفاجئ بسبب اعتمادها على شجرة واحدة فقط، كما يعاني من مشكلة عنق الزجاجة bottleneck عند مصدر الفيديو [33].

7.4.4. التطبيق زيك زاك

يُعدّ التطبيق زيك زاك من أفضل تطبيقات البث، ويتألف من مخدم وسائط وحيد يبث الفيديو إلى عدد من المستخدمين المنظمين ضمن بنية شجرية [40].

أ. بنية التطبيق:

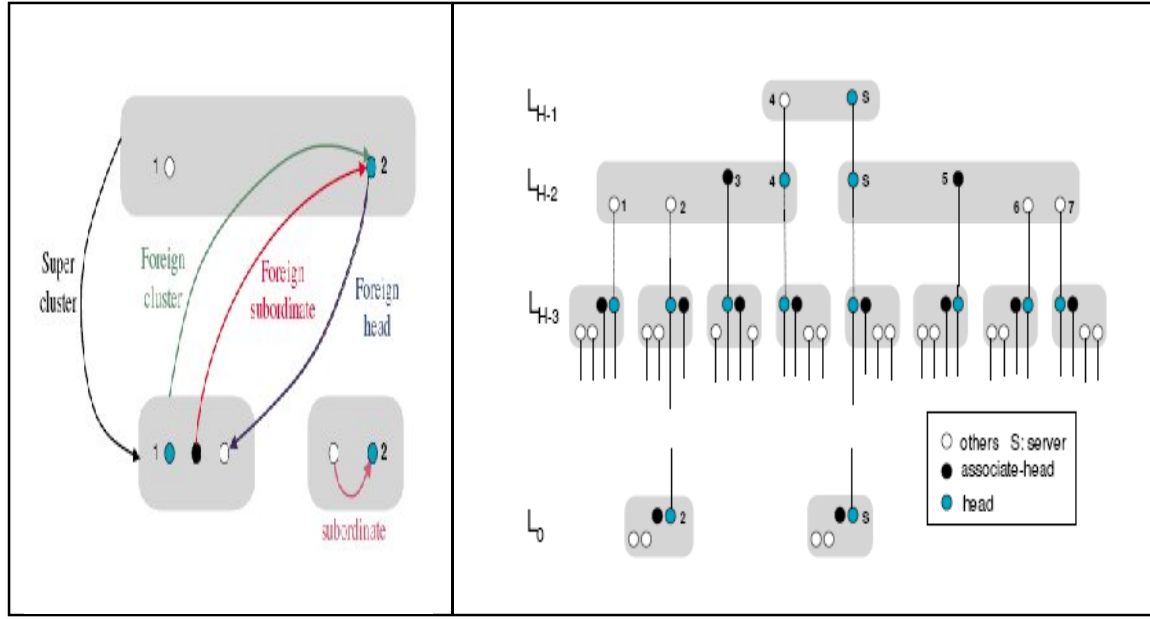
تتألف من مكونين entities أساسيين هما [40]:

- التنظيم الإداري administrative organization: ويمثل العلاقات التراكبية بين العقد.
- شجرة البث المتعدد Multicast tree: وتمثل العلاقات الفيزيائية بين العقد (كيفية اتصال العقد مع بعضها البعض لتبادل المحتويات).

ب. التنظيم الإداري:

يجري تنظيم العقد ضمن بنية هرمية متعددة الطبقات multi-layer hierarchy، وفقاً للآتي (بعد H عدد الطبقات و $k > 3$ عدد ثابت) كما هو مبين بالشكل (4-4):

- الطبقة 0 تضم جميع العقد في المنظومة.
- العقد في الطبقة $H-1$ $z < H-1$ تقسم إلى عناقيد بحجم ضمن المجال $(2, 3k)$.
- الطبقة $H-1$ تضم عنقود واحد بحجم ضمن المجال $(2, 3k)$.
- يجري اختيار أحد العقد في كل عنقود من الطبقة $j-1$ ليكون رأساً للعنقود head، ومن ثمّ يكون عنصراً ضمن الطبقة j ، كما يجري اختيار رأس مساعد associated head [40].



(ب)

(أ)

الشكل (4-4) (أ) التنظيم الإداري للعد، (ب) العلاقة بين العقد والعناقد [40].

ج. شجرة البث المتعدد:

نبين فيما يأتي المصطلحات الضرورية لبناء شجرة البث المتعدد:

• المرؤوس (x) subordinate: العقد غير الرئيسية non head في العنقود والتي تكون مرؤوسة من قبل الند x.

• الرأس الخارجي foreign head: العقد غير الرئيسية المرافقة لـ x في الطبقة $j > 0$

• المرؤوس الخارجي foreign subordinate: العناصر التابعة لـ x في الطبقة $j-1$

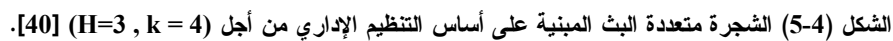
• العنقود الخارجي foreign cluster: العنقود المتضمن لـ x في الطبقة $j-1$

أما بناء الشجرة فيجري اعتماداً على مجموعة قواعد تدعى C-rules وهي:

- القاعدة الأولى: الند غير الموجود في طبقته العليا لا يملك أية وصلات داخلية أو خارجية.

- القاعدة الثانية: العقد غير الرئيسية في العنقود يجب أن تملك وصلات مباشرة من الرأس المساعد

- القاعدة الثالثة: الرأس المساعد في العنقود (عدا المخدم s) يجب أن يملك وصلة من الرأس الخارجي [40].



ثمة عدة ميزات للتطبيق زيك زاك من أهمها:

- ## 5.4. الخلاصة

71

تضمن الفصل تصنيفاً لتطبيقات الند للند من أجل بث الفيديو مع أمثلة عملية على كل منها، إذ تقسم تبعاً لطريقة توضع العقد ضمن الشبكة إلى منظومات بنيوية، تعتمد على تنظيم العقد ضمن بنية أحادية الشجرة أو متعددة الأشجار، ومنظومات غير بنيوية، حيث تتوضع العقد بشكل عشوائي، إضافة إلى منظومات هجينة تمتلك خصائص كلا المنظومتين.

من جهة أخرى يمكن تصنيف هذه التطبيقات تبعاً لبنية المنظومة إلى منظومات تعتمد على مخدم مركزي مسؤول عن تنظيم العقد وإدارة الشبكة، ومنظومات موزعة حيث تكون عمليات التنظيم والإدارة موزعة على العقد، وكذلك إلى منظومات هرمية تكون فيها العقد منظمة على شكل بنية هرمية.

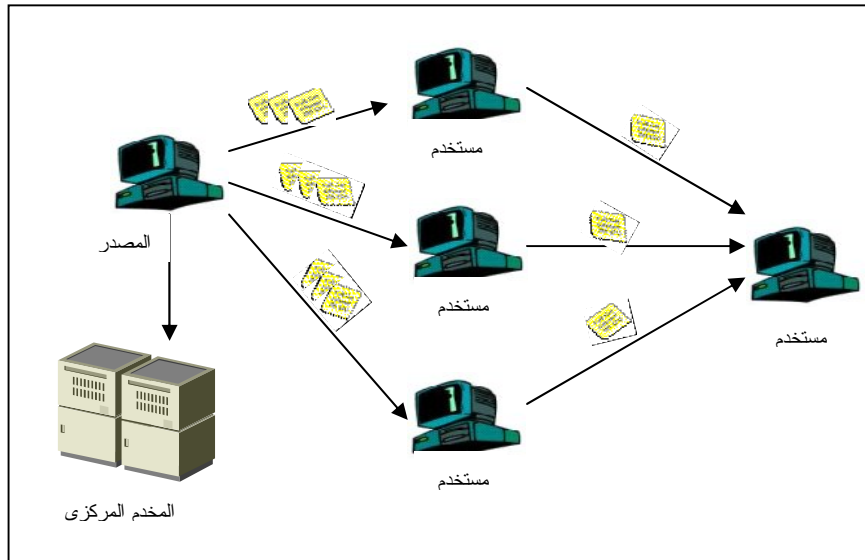
كما استعرضنا في نهاية الفصل بعض أهم التطبيقات مثل التطبيق نايس، والتطبيق سبلت ستريم، والتطبيق ناردا، والتطبيق بوليت، والتطبيق كوبنت، والتطبيق سبريد إيت، والتطبيق زيك زاك.

نموذج التطبيق المقترح باستخدام شبكات الند للند

يعرض هذا الفصل للنموذج المقترح لتطبيق ند لند مستخدم من أجل مشاركة الملفات وكذلك البث الفيديوي في الزمن الحقيقي. سنعتمد في بناء التطبيق على البنية الهجينة لشبكات الند للند، والتي تعتمد على وجود مخدّم مركزي لإدارة الملفات وبناء الشبكة التراكيبية بين العقد. يشتمل هذا الفصل على البنية المقترحة للتطبيق، وآلية اتصال العناصر المكونة لها مع بعضها البعض، والآلية المعتمدة لاختيار العقد من أجل بناء الشبكة. كما يوضح بنية الحاوية container المستخدمة لإرسال كتل الملف عبر الشبكة إلى المستخدمين، والآلية المعتمدة لتقسيم الملف إلى كتل في المرسل وإعادة ترتيبها وتجميعها في المستقبل، إضافةً إلى آلية إعادة بناء الشبكة بسرعة وكفاءة، في حال فشل بعض العقد أو مغادرتها الشبكة بشكل مفاجئ.

1.5. البنية المقترحة للمنظومة

تمثل البنية المقترحة لتطبيق ند لند هجين يعمل بحالتين وهما، مشاركة الملفات وبث الفيديو بالزمن الحقيقي لعدد من المستخدمين. وتتألف البنية المقترحة، كما هو مبين بالشكل (1-5)، من مخدّم مركزي مسؤول عن تخزين معلومات الملفات المراد مشاركتها، وعن إدارة اتصال العقد وتنظيمها، ومن المصدر source الذي يملك ملف أو أكثر ويرغب بمشاركتها، وأخيراً من مستخدمين يستقبلون الملف من المصدر ويعيدون إرساله إلى مستخدمين آخرين، أي أنها تعمل كعناصر مُرَجِّلة (نقل على مراحل) relay للملف.



الشكل (1-5) البنية المقترحة للتطبيق.

تمتاز تطبيقات البث الفيديوي بأن حجم الفيديو كبير بحيث من الصعب تأمينه من قبل مصدر واحد فقط، ولذلك اعتمدنا في بناء التطبيق على تقسيم ملف الفيديو إلى أجزاء بحجم ثابت، حيث يكون كل مصدر مسؤول عن إرسال جزء أو أكثر من ملف الفيديو تبعاً لعرض المجال المتاح له. ويقوم المستخدم بدوره باستقبال أجزاء الفيديو المختلفة وإعادة ترتيبها للحصول على الفيديو كاملاً. وبيّن الجدول (5-1) مقارنة بين بنية التطبيق المقترح وبعض التطبيقات الشهيرة.

الجدول (5-1) مقارنة بين التطبيق المقترح وبعض التطبيقات الشهيرة.

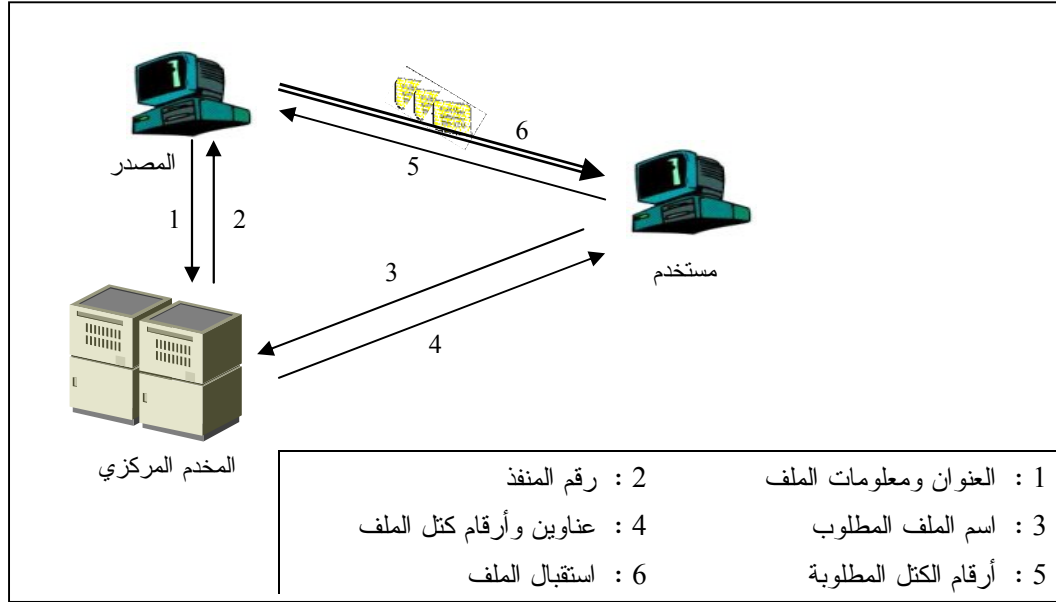
التطبيق نابستر	التطبيق غوتلا	التطبيق بت تورنت	التطبيق المقترح
مخدم مركزي	لا يوجد مخدم	مخدم مركزي (متعقب)	مخدم مركزي
عملية بحث فعالة وسريعة	تكلفة بحث موزعة	—	عملية بحث سريعة وفعالة
—	—	تقسيم الملف إلى أجزاء متساوية (كتل)	تقسيم الملف إلى أجزاء متساوية بالحجم (كتل)
—	تحميل زائد زيادة حجم المعالجة	—	—

2.5. آلية عمل التطبيق

عند انضمام مصدر جديد إلى الشبكة فإنه يُعرف عن نفسه للمخدم بواسطة إرسال العنوان الخاص به، وكذلك معلومات عن الملفات التي يمتلكها. ومن هذه المعلومات اسم الملف وحجمه. يقوم بعدها المخدم بتخزين هذه المعلومات ضمن قاعدة بياناته، ثم يخصص رقم منفذ وحيد لكل ملف، ويعيد إرساله إلى المصدر الذي يقوم باستخدامه من أجل استقبال طلبات المستخدمين. يجري في المصدر تقسيم الملف إلى كتل بحجم ثابت، توضع كل منها ضمن حاوية تضم معلومات عن الرقم التسلسلي sequence number للكتلة، وزمن إرسالها sending time. ويستفاد من المعلومات المتضمنة بالحاوية عند إعادة تجميع كتل الملف في المستقبل.

عند انضمام مستخدم جديد إلى التطبيق، فإنه يقوم بإرسال استفسار باسم الملف المطلوب إلى المخدم، الذي يقوم بدوره بإعادة إرسال عناوين المصادر والعقد المناسبة التي يستطيع المستخدم الاتصال بها للحصول على الملف المطلوب، حيث يجري اختيار العقد المناسبة تبعاً للمخطط

التدفقي المبين في الشكل (5-6). يقوم المستخدم فيما بعد بالاتصال بهذه العقد، واستقبال كتل الملف منها، ومن ثم إعادة تجميعها للحصول على كامل الملف. ويوضح الشكل (5-2) آلية عمل هذا التطبيق.



الشكل (5-2) آلية عمل التطبيق المقترح.

3.5. مكونات المنظومة

تتألف البنية المقترحة للتطبيق من ثلاثة مكونات أساسية هي: المخدم المركزي، والمصدر، والزبون.

1.3.5. المخدم المركزي

يتمثل الهدف الأساسي من وجود المخدم المركزي في تنظيم العقد ضمن المنظومة بسهولة وكفاءة، وكذلك تخزين معلومات الملفات وعناوين العقد التي تمتلكها مما يسهل عملية البحث. عند اتصال مصدر الملف بالمخدم، فإنه يرسل عنوانه ومعلومات الملف الذي يرغب بمشاركته، عندها يقوم المخدم بتخزين هذه المعلومات ضمن قاعدة البيانات الخاصة به، والتي تتألف من الحقول الآتية: مُعين الملف (ID)، واسم الملف، وحجم الملف، وعنوان المصدر (IP)، ورقم المنفذ الذي يقوم المخدم بتخصيصه للملف، كما هو مبين بالشكل (5-3).

ID	HOST NAME	FILE NAME	FILE SIZE	Port

الشكل (3-5) بنية قاعدة البيانات في المخدم.

كما يتضمن المخدم جدولاً ثانياً يدعى جدول التخصيص allocation table، وفيه يخصص لكل ند سطر يضم المعلومات الآتية:

- عنوان الند والذي يمثل مفتاح الجدول IP
 - العدد الكلي للحيزات slots ضمن إطار الوصلة TL link
 - العدد الكلي للحيزات ضمن إطار المحتوى TS content
 - عناوين العقد التي يقوم الند بتخديمها PC
 - شعاع تخصيص الحيزات slot allocation وهو عبارة عن أزواج من رقم الحيز للمحتوى مع رقم الحيز المخصص لها من الوصلة VC (link slot, content slot)
- ويبين الشكل (4-5) بنية جدول التخصيص في المخدم.

IP	TL	TS	PC				VC			

الشكل (4-5) بنية جدول التخصيص في المخدم.

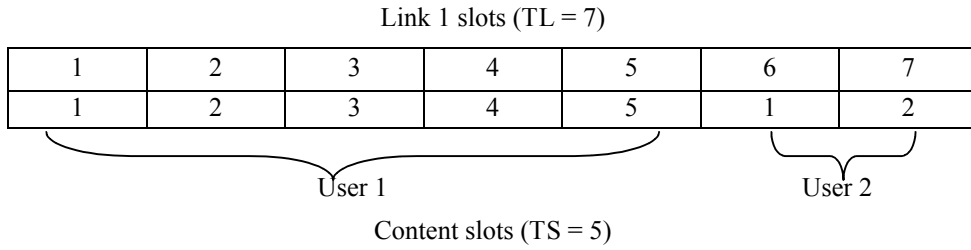
يقوم المخدم بتعبئة هذا الجدول اعتماداً على آلية مطورة لهذه الغاية ومبيّنة في الفقرة (4.5). عند استفسار أي زبون عن اسم ملف فإن المخدم يقوم بالبحث ضمن قاعدة بياناته، ثم يرسل للزبون عناوين العقد المناسبة للاتصال بها، وكذلك كتل الملف المتاحة في كل عقدة من العقد، وعلى أي حيز من الوصلة يجب على المستخدم استقبالها.

2.3.5. المصدر

يمتلك المصدر ملف أو أكثر ويرغب بمشاركتها مع باقي مستخدمي التطبيق. في البداية يُعرف المصدر عن نفسه للمخدم بواسطة إرسال عنوانه ومعلومات الملف التي يملكها، ثم يستقبل من المخدم رقم المنفذ الواجب استخدامه من أجل كل ملف من الملفات.

يقوم المصدر بعدها بفتح المنفذ وانتظار طلبات المستخدمين، ويقوم بنفس الوقت بتهيئة الملف حيث يقوم بتقسيمه إلى كتل بحجم ثابت، ويضع كل كتلة ضمن حاوية مصممة لهذا الغرض. تُعد بنية الحاوية المقترحة بسيطة جداً حيث تتضمن حقلين أساسيين هما: الرقم التسلسلي للكتلة، والذي يستخدم من أجل إعادة ترتيب وتجميع الكتل بشكل صحيح عند الاستقبال، وزمن إرسال الكتلة بحيث يستخدم من أجل تحديد التأخير، ومن ثمّ مزامنة الكتل وتجميعها بشكل صحيح في المستقبل.

عند ورود طلب من المستخدم فإن المصدر يقوم بوضع كتل الملف ضمن حيزات الوصلة المناسبة لطلبات المستخدم، ثم إعادة إرسالها إليه. قد تتضمن الوصلة الواحدة كتل ملف من أجل عدة مستخدمين حيث يقع على المصدر مهمة تخصيص حيزات الوصلة المناسبة لكل مستخدم من المستخدمين، كما هو موضح بالشكل (5-5).



الشكل (5-5) تخصيص حيزات الوصلة من أجل حيزات المحتوى.

3.3.5. المستخدم

عندما يرغب المستخدم بالحصول على ملف ما، يرسل استفساراً باسم الملف إلى المخدم الذي يقوم بإرسال عناوين المصادر والعقد الموجودة ضمن الشبكة التي يمكن للمستخدم الاتصال بها، وكذلك أرقام كتل الملف وحيزات الوصلة المخصصة لها، وذلك من أجل كل مصدر أو عقدة مرسل.

عندها يقوم المستخدم بالاتصال بهذه المصادر والعقد، واستقبال كتل الملف المطلوبة من كل منها، وذلك ضمن حيزات الوصلة المخصصة لها. بعدها يقوم المستخدم بإعادة ترتيب هذه الكتل بالاستفادة من معلومات الحاوية، ثم تخزينها ضمن ملف بحالة مشاركة الملفات أو عرضها بعد إجراء عمليات المعالجة المطلوبة مثل فك الترميز ومزامنة الصوت والصورة في حالة البث الفيديوي.

بعد استقبال المستخدم عدداً من كتل الفيديو يؤدي دور المرسل، حيث يقوم بإعادة إرسال هذه الكتل إلى مستخدمين آخرين موجودين ضمن الشبكة.

4.5. آلية تنظيم العقد ضمن الشبكة

المخدم المركزي هو المسؤول عن تنظيم العقد ضمن الشبكة تبعاً لآلية محددة. وسنقوم فيما يأتي بتوضيح بعض المفاهيم والرموز المستخدمة لهذه الغاية.

تجري عملية تخصيص العقد من أجل كل ملف والذي سنرمز له بـ (c)، علماً أن كل محتوى مرتب ضمن أطر كل منها مقسم إلى عدد من الحيزات والتي يرمز لعددتها بـ TS.

يجري تعريف كل مكون (مصدر أو عقدة) بالعنوان الخاص به IPC_i . ومن أجل كل مكون فإنه يجري تخصيص السطر i-th من الجدول، والذي يتضمن المعلومات الآتية:

- TL_i العدد الكلي للحيزات ضمن كل إطار من أطر الوصلة

- PC_{ij} شعاع يتضمن عناوين العقد التي يجري تخدمها بواسطة المكون i، حيث يشير j إلى العقد المتصلة مع المكون i.

- VC_{ijk} شعاع يضم أزواج من حيز وصلة مع حيز محتوى، حيث يشير إلى تخصيص حيز المحتوى ضمن حيز وصلة محددة. علماً أن k يتراوح بين الواحد وعدد الحيزات المخصصة.

- M عدد حيزات الوصلة المخصصة

- TF عدد حيزات الوصلة المتاحة

- المخطط التدفقي لآلية تنظيم العقد ضمن الشبكة

في المرحلة الأولى نقوم بحساب العدد الإجمالي لحيزات الوصلة TL وحيزات المحتوى TS وذلك اعتماداً على عرض المجال لكل من الوصلة BWL والمحتوى BWS، وفقاً للعلاقتين (1-5) و (2-5):

$$TL = \frac{BWL}{Bunit} \quad (5-1), \quad TS = \frac{BWS}{Bunit} \quad (5-2)$$

حيث يمثل Bunit وحدة عرض المجال، وهي قيمة افتراضية تستخدم لتبسيط العمل.

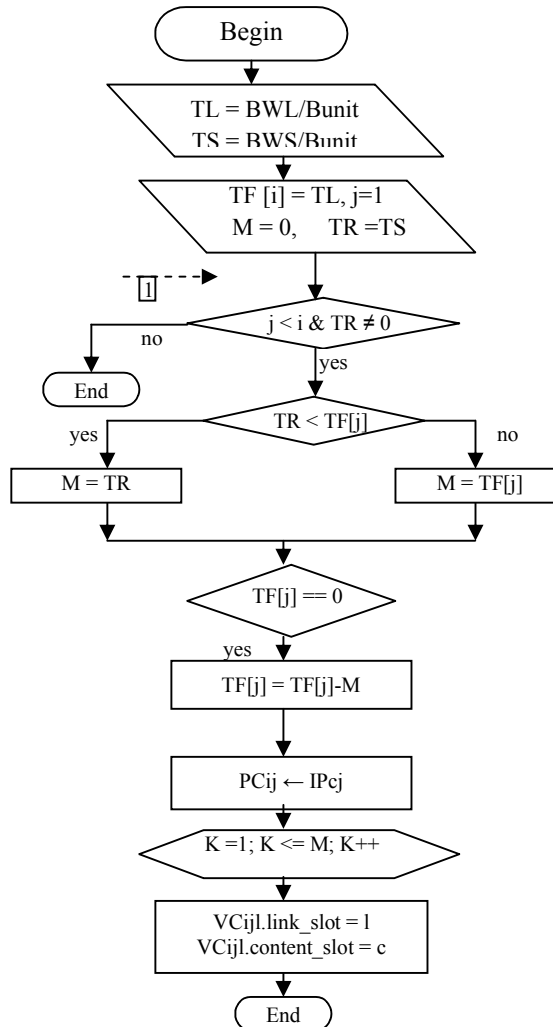
في المرحلة الثانية ومن أجل كل طلب جديد فإننا نقوم بتحديد عدد حيزات الوصلة المتاحة لكل عقدة بدءاً من العقدة الأولى وحتى العقدة (i-1) حيث يشير i إلى العقدة الأخيرة التي انضمت إلى الشبكة. في حال وجود حيزات وصلة متاحة ضمن العقدة j فإنه يتم إضافة عنوان هذه العقدة IPC_j إلى شعاع العناوين التي تُخدّم من قبل الند i أي ضمن شعاع PC_{ij} ، وهو ما يعبر عنه بالعلاقة (3-5):

$$\text{if } (TF_j \neq 0) \quad \text{then} \quad PC_{ij} = IPC_j \quad (5-3)$$

وأخيراً نقوم بتخصيص حيزات المحتوى إلى حيزات الوصلة المتاحة ضمن الشعاع VC_{ijk} كما هو مبين في العلاقة (4-5).

$$\text{for } k = 1 \text{ to } M \\ VC_{ijk} = (l, c) \quad (5-4)$$

حيث يشير كل من l و c إلى دليل حيز الوصلة ودليل حيز المحتوى على الترتيب. ويبين الشكل (5-6) المخطط التدفقي لآلية بناء الشبكة.



الشكل (5-6) المخطط التدفقي لآلية بناء الشبكة.

5.5. آلية تقسيم الملف وإعادة تجميعه

نظراً للحجم الكبير لبعض أنواع الملفات وخاصة ملفات الفيديو، لا يمكن الاعتماد على ند واحد من أجل عملية الإرسال، لذلك لابد من تقسيم الملف إلى أجزاء يتم إرسالها من قبل عدة عقد إلى المستقبل، الذي تقع عليه مهمة إعادة ترتيبها وتجميعها للحصول على كامل الملف. تقع عملية تقسيم الملف على عاتق المصدر حيث يقوم بتقسيمه إلى كتل بحجم ثابت، ثم يقوم بوضع كل كتلة ضمن حاوية تتضمن الرقم التسلسلي للكتلة والذي يحدد ترتيب الكتل ضمن الإطار، إضافة إلى حقل يضم ترتيب الإطار ضمن المحتوى. ويبين الشكل (5-7) بنية الحاوية.

Sequence	Frame order	Data
----------	-------------	------

الشكل (5-7) بنية الحاوية المقترحة.

يلي عملية التقسيم تخصيص الكتل ضمن حيزات الوصلة المناسبة وفقاً للمخطط التدفقي المذكور سابقاً ومن ثم إرسالها إلى المستخدمين. بعد استقبال كتل الملف يقوم المستخدم بإعادة ترتيب هذه الكتل وتجميعها بالشكل الصحيح. ويحتاج من أجل ذلك إلى ذاكرة مؤقتة بحجم مساوي لحجم الإطار أي $(BWS = TS * \text{Bunit})$. عند استقبال الكتل من عدة مرسلين، يقوم المستخدم بقراءة معلومات الحاوية لتحديد ترتيب الكتل ضمن الإطار حسب الرقم التسلسلي للكتلة. عندها يتم ترتيب الكتل ضمن الذاكرة المؤقتة بحيث تصبح جاهزة لعملية العرض وكذلك لإعادة إرسالها إلى عقد أخرى.

6.5. آلية إعادة بناء الشبكة

تعاني شبكات الند للند من ديناميكية عالية للعقد حيث يمكن أن تنضم إلى الشبكة وتغادرها بسرعة وبدون إعلام مسبق. لذلك كان من الضروري إيجاد آلية لإعادة بناء الشبكة بسرعة وكفاءة في حال فشل بعض العقد أو مغادرتها. عند فشل مستخدم ما من استقبال كمية محددة من كتل الملف من مصدر أو مرسل معين، فإنه يقوم بإعلام المخدم، وذلك بإرسال رسالة تتضمن عنوان العقدة التي غادرت الشبكة، إضافة إلى اسم الملف الذي يقوم الند بمشاركته.

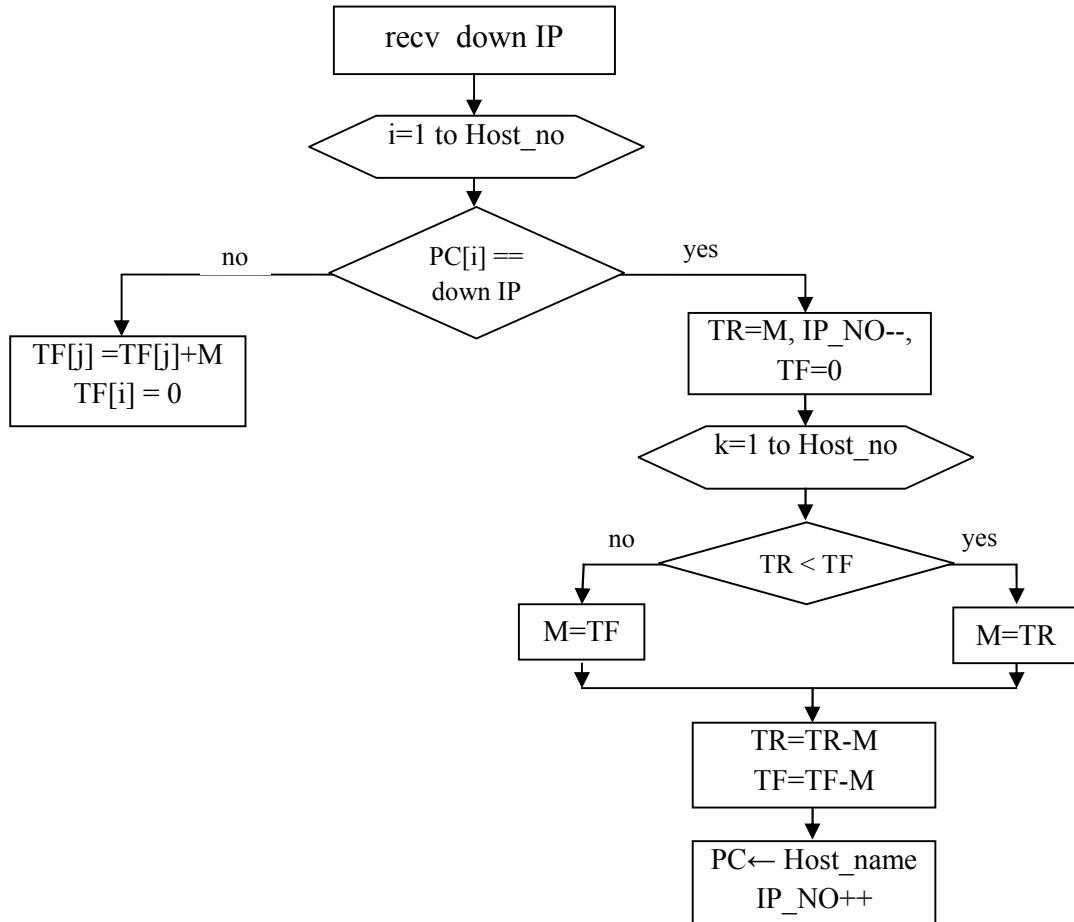
عند استقبال المخدم لرسالة الإعلام هذه، فإنه يستخدم اسم الملف من أجل الوصول إلى جدول التخصيص المطلوب، حيث يوجد جدول تخصيص محدد لكل ملف من الملفات المشاركة ضمن الشبكة.

بعد تحديد جدول التخصيص المناسب يقوم المخدم بالبحث ضمن حقل العناوين IPc عن السطر الذي يوافق عنوان العقدة الفاشلة fail node، ومن ثم تحرير حيزات الوصلات التي كانت مخصصة لهذه العقدة. ويبيّن الشكل (8-5) آلية البحث هذه.

```
for i = 1 to Host_No
    if (IP[i] == "fail IP" )
        TF[i] = TF[i] + M[i ]
```

الشكل (8-5) آلية البحث عن العقدة الفاشلة.

ثم يقوم المخدم بالبحث عن عنوان العقدة الفاشلة ضمن حقل العناوين المُخدمة PC لتحديد العقد التي كانت مُخدمة من قبل هذه العقدة الفاشلة، ثم تُجرى عملية تخصيص جديدة لاستبدال العقدة الفاشلة بعقد جيدة وفقاً للمخطط التدفقي المبين في الشكل (9-5).



الشكل (9-5) المخطط التدفقي لإعادة بناء الشبكة عند فشل عقدة ضمنها.

7.5. التنفيذ البرمجي للتطبيق

استخدمنا لتنفيذ التطبيق برمجياً لغة سي C، وذلك لكتابة ثلاثة برامج أساسية لتمثيل كل من المخدم، المسؤول عن تنظيم الشبكة، والمصدر الذي يمتلك الملف، والمستخدم. اعتمدنا في التعامل مع الفيديو على مكتبة مفتوحة تدعى ffmpeg، وهي عبارة عن مكتبة تضم برامج سي C مسؤولة عن عمليات ترميز الفيديو وفك ترميزه، واستخلاص رزم الصورة والصوت، وغيرها من عمليات التحويل بين الأنماط المختلفة للصورة والصوت. استخدمنا لعرض الفيديو على الشاشة مكتبة خاصة تدعى Simple Direct Library (SDL)، وهي مكتبة مفتوحة تتألف من عدة برامج مكتوبة بلغة سي تتيح عرض الفيديو على الشاشة والتحكم به.

1.7.5. المكتبات البرمجية المستخدمة

استخدمنا عدة مكتبات برمجية لبناء التطبيق وهي مبينة بالجدول (2-5).

الجدول (2-5) المكتبات البرمجية المستخدمة

المكتبة	استخدامها
stdio.h	عمليات الإدخال والإخراج على الشاشة
stdlib.h	عمليات إظهار الأخطاء والتحكم بها
netinet/in.h	تضم البنى الخاصة بعنوان المقابس
sys/type.h	تضم أنواع المقابس
sys/socket.h	عمليات التعامل مع المقبس والتحكم به
netdb.h	التحويل بين أسماء المضيفين وعناوينهم
arpa/inet.h	
libavformat/avformat.h	عمليات التعامل مع الفيديو
libavcodec/avcodec.h	معلومات الترميز والتعامل معها
sdl.h	عمليات العرض على الشاشة والتحكم بالصوت

2.7.5. التنفيذ البرمجي للمخدم

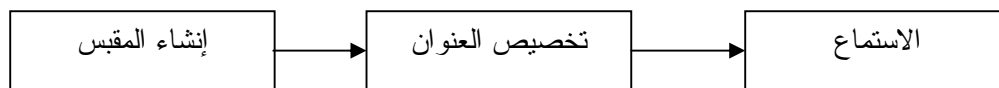
يُعدّ المخدم مسؤولاً عن تنظيم اتصال العقد مع بعضها البعض وعن عملية تخزين الملفات والتحكم بعملية البحث.

من الممكن تقسيم برنامج المخدم إلى أربعة أقسام أساسية وهي: تهيئة الاتصال، والتعامل مع المصدر، والتعامل مع المستخدم، وإعادة بناء الشبكة عند الفشل.

أ- تهيئة الاتصال

في البداية، قمنا بإنشاء مقبس socket من النمط الدفقي sock_stream، وخصصنا له عنوان وهو "192.168.1.5"، ورقم منفذ مساوي لـ 7000، من ثم جعلناه بوضع الاستماع listening وذلك لتلقي طلبات المستخدمين.

وبيّن الشكل (5-10) المخطط الصندوقي لآلية تهيئة الاتصال، وبيّن الشكل (5-11) الخطوات البرمجية لتحقيق ذلك.



الشكل (5-10) آلية تهيئة الاتصال

```

//socket creation
int socket_master = socket(AF_INET, SOCK_STREAM, 0);
//define server information
server.sin_family = AF_INET; //server type (Internet)
server.sin_port = htons(atoi(PORT)); //server port (7000)
server.sin_addr.s_addr = inet_addr(IP); //server IP (192.168.1.5)
//bind IP address and port to socket
bind(socket_master, (struct sockaddr *) &server, address_size);
//listen to clients
listen(socket_master, 3);
  
```

الشكل (5-11) الخطوات البرمجية لتهيئة الاتصال

ب- التعامل مع المصدر

عند ورود طلب من المصدر إلى المخدم، يجب أن يتضمن الطلب معلومات عن عنوان المصدر HOST_NAME، والملف المراد مشاركته. ومن هذه المعلومات اسم الملف FILE_NAME، وحجمه FILE_SIZE، ومعدل البتات BIT_RATE.

بعدها يقوم المخدم بإضافة هذه المعلومات إلى قاعدة بياناته، ثم تخصيص رقم منفذ للتعامل مع الملف من خلاله وكذلك مدة الملف Duration. اخترنا رقم المنفذ بدءاً من 7000 وفقاً للمعادلة (5-5).

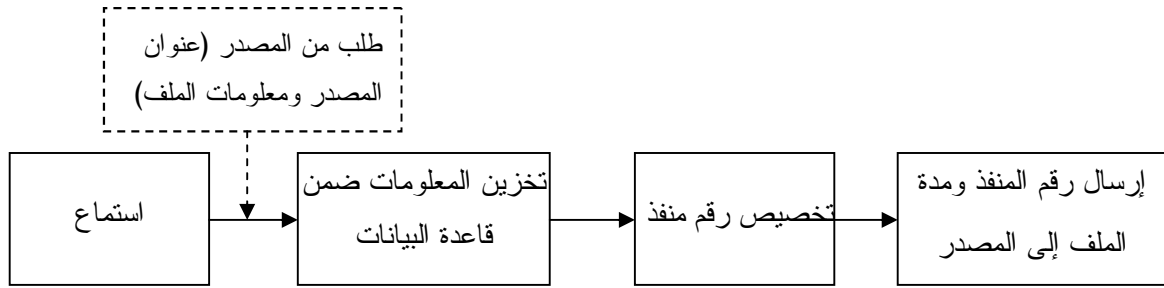
$$\text{port} = \text{default_port} + \text{ID};$$

(5-5)

since: default_port = 7000, ID = number of file in the data base

وذلك لتجنب أرقام المنافذ محددة والتي تتراوح بين (1-1024).

يبين الشكل (5-12) المخطط التدفقي لتعامل مع المصدر، أما الشكل (5-13) فيبين برنامج سي الذي يتعامل مع المصدر.



الشكل (5-12) المخطط التدفقي من أجل التعامل مع المصدر

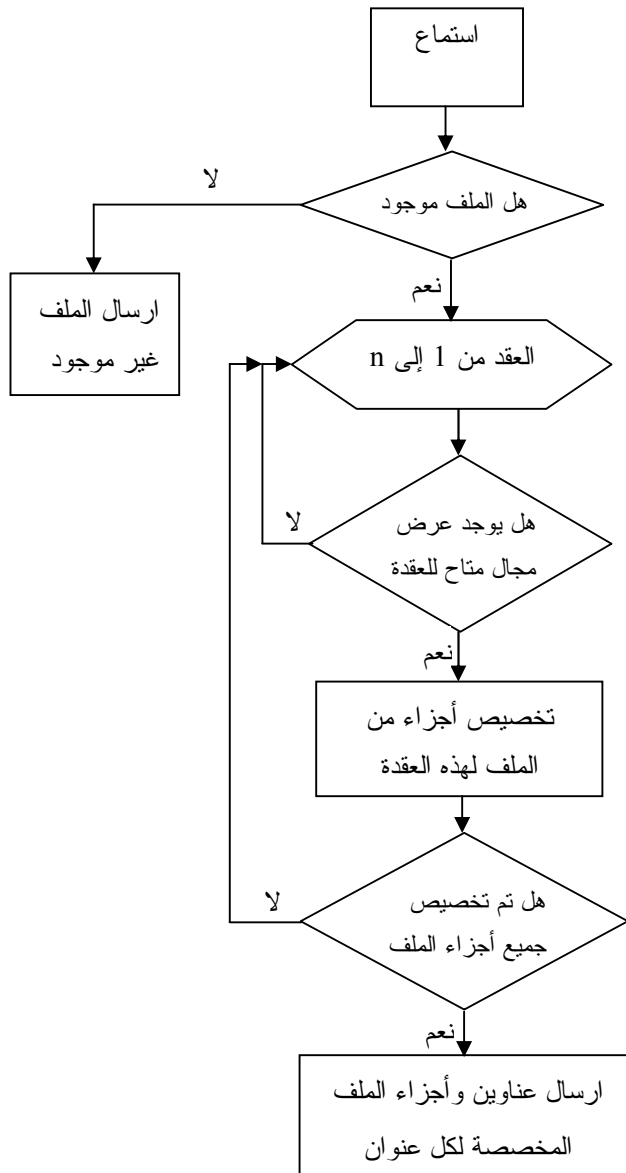
```
//accept new connection
int socket_new = accept(socket_master, (struct sockaddr *) &client,
&client_length);
//receive file information
recv(socket_new, buffer, BUFFER_SIZE, 0);
//insert new item in data base
Data_base[file_no+1].HOST_NO = 1;
sscanf(buffer, "%s\t%s\t%d\t%d\n", &HOST_NAME,
&FILE_NAME, &FILE_SIZE, &BIT_RATE);
for(i=1; i <= file_no; i++){
    file_no++; qfile_no = file_no;
    Data_base[file_no].ID = file_no;
    Data_base[file_no].PORT= Default_port + Data_base[file_no].ID;
//Default_port = 2000
strcpy(Data_base[file_no].HOST_NAME[Data_base[file_no].HOST_NO],
HOST_NAME);
strcpy(Data_base[file_no].FILE_NAME, FILE_NAME);
Data_base[file_no].BIT_RATE = BIT_RATE;
Data_base[file_no].DURATION = FILE_SIZE/BIT_RATE;}
//send port & file duration to use for streaming
sprintf(buffer, "%d\t%d\n",
Data_base[qfile_no].PORT, Data_base[file_no].DURATION);
send(socket_new, buffer, sizeof(buffer), 0);
```

الشكل (5-13) برنامج سي من أجل التعامل مع المصدر

ج- التعامل مع المستخدم

عند ورود طلب من المستخدم إلى المخدم، ويتضمن اسم الملف المراد الحصول عليه. عندها يقوم المخدم بالبحث عن اسم الملف ضمن قاعدة بياناته، في حال وجود الملف فإنه يقوم بتحديد عناوين المصادر والعقد التي تملك الملف ويوزع على كل منها أجزاء الملف المسؤول عن إرسالها إلى المستخدم، وذلك تبعاً لعرض المجال المتاح لكل عقدة.

يوضح الشكلان (14-5) و (15-5) آلية عمل المخدم عند ورود طلب من المستخدم وبرنامج سي المكتوب لهذا الغرض.



الشكل (14-5) المخطط التدفقي لعمل المخدم عند ورود طلب من المستخدم

برنامج المستخدم

```
strcpy(info,"n");
printf("client: receive request file information\n");
//receive file name
bzero(buffer,BUFFER_SIZE);
recv(socket_new, buffer, BUFFER_SIZE, 0);
sscanf(buffer,"%s\t%s\n",&FILE_NAME,&HOST_NAME);
printf("query for %s\n",FILE_NAME);
//query for file name
qfile_no = 0;
for(i=1; i <= file_no; i++){
    if(strcmp(Data_base[i].FILE_NAME,FILE_NAME) == 0) qfile_no = i;}
    if(qfile_no != 0) strcpy(info,"y");
    send(socket_new, info ,1,0);
//insert client IP into Data Base
Data_base[qfile_no].HOST_NO++;
strcpy(Data_base[qfile_no].HOST_NAME[Data_base[qfile_no].HOST_NO],HOST_NAME);
//send port number to client and file duration
sprintf(buffer, "%d\t%d\n",
Data_base[qfile_no].PORT,Data_base[qfile_no].DURATION);
send(socket_new, buffer, sizeof(buffer), 0);
printf("send the port %d to client\t
Duration : %d\n",Data_base[qfile_no].PORT,Data_base[qfile_no].DURATION
);
//determine available source IP addresses
allocation(qfile_no);
//send source IP addresses to client
send(socket_new&allocate_table1[qfile_no],sizeof(allocate_table1[qfile_no]),0);}}
```

الشكل (5-15) برنامج سي من أجل التعامل مع المستخدم.

د- إعادة بناء الشبكة عند الفشل

عند ورود إعلام بفشل عقدة ضمن الشبكة إلى المخدم فعندها نناقش ثلاث حالات للعقدة الفاشلة:

- العقدة عبارة عن مستخدم يستقبل الملف دون إعادة إرساله، عندها نقوم بعملية تحرير الوصلات التي كانت تزود هذه العقدة بالملف.
- العقدة عبارة عن مصدر مسؤول عن إرسال الملف، عندها نقوم بإعادة تخصيص لعناوين وأجزاء الملف.
- العقدة عبارة عن مستخدم يستقبل الملف ويرسله أيضاً، عندها نقوم بعملية تحرير للوصلات وإعادة تخصيص جديدة.

يبين الشكل (5-16) البرنامج المكتوب بلغة سي لتنفيذ هذه الآلية.

```
// receive IP addresses down and file name
recv(socket_new,buffer,sizeof(buffer),0);
sscanf(buffer,"%s\t%s\n",&FILE_NAME,&HOST_NAME);
// search on file name
for(i =1; i <= file_no; i++) if(strcmp(Data_base[i].FILE_NAME,FILE_NAME)
== 0) qfile_no = i;
// new allocation
for(i=1; i<=Data_base[qfile_no].HOST_NO; i++)
for(j=1;j<=Data_base[qfile_no].HOST_NO;j++){
    if(strcmp(allocate_table1[qfile_no].PC[i][j],"NO")!=0){
        if(strcmp(allocate_table1[qfile_no].PC[i][j],HOST_NAME)==0)
new_allocation(qfile_no,i,j);
        else if(strcmp(allocate_table1[qfile_no].HOST_NAME[i],HOST_NAME)==0)
free_alloc(qfile_no,i,j);
    }}

// new allocation
void new_allocation(int file_no, int i, int j){
TR = allocate_table1[file_no].M[i][j];
allocate_table1[file_no].IP_NO[i]--;
allocate_table1[file_no].TF[j] = 0;
while(k <= Data_base[file_no].HOST_NO && TR != 0){
    allocate_table1[file_no].M[i][k] = (TR < allocate_table1[file_no].TF[k])
? TR : allocate_table1[file_no].TF[k];
    TR = TR - allocate_table1[file_no].M[i][k];
    allocate_table1[file_no].TF[k] = allocate_table1[file_no].TF[k] -
allocate_table1[file_no].M[i][k];

strcpy(allocate_table1[file_no].PC[i][k],Data_base[file_no].HOST_NAME[k])
;
    allocate_table1[file_no].IP_NO[i]++;
}}

// free allocation
void free_alloc(int file_ID,int i, int j){
allocate_table1[file_ID].TF[j] = allocate_table1[file_ID].TF[j] +
allocate_table1[file_ID].M[i][j];
```

الشكل (5-16) برنامج سي المسؤول عن إعادة بناء الشبكة عند فشل أحد العقد

3.7.5. التنفيذ البرمجي للمصدر

يملك المصدر ملف واحد أو أكثر ويرغب بمشاركتها عبر الشبكة. في البداية، يقوم المصدر بالاتصال مع المخدم، ومن ثم يرسل إليه عنوانه ومعلومات الملف الذي يرغب بمشاركته. يجري الحصول على معلومات الملف في حالة الفيديو بالاستفادة من توابع مكتبة الـ ffmpeg ومن هذه المعلومات: اسم ملف الفيديو، وحجمه، وعدد بتاته.

بعدها يجري استقبال كل من رقم المنفذ المخصص للملف ومدته، حيث يقوم المصدر باستخدام رقم المنفذ لتهيئة الاتصال والاستماع إلى طلبات المستخدمين. يبين الشكل (5-17) برنامج سي المخصص للاتصال بالمخدم.

```

int socket_master = connection(argv[1], argv[2]);

//enter video info
printf("Enter file name      \t:\t"); scanf("%s", &file1_info.FILE_NAME);
printf("Enter host name      \t:\t"); scanf("%s", &file1_info.HOST_NAME);
printf("Enter bit rate (kb) \t:\t"); scanf("%d", &BIT_RATE);
file1_info.BIT_RATE = BIT_RATE*Bunit;

//video initialization
av_register_all();

//video information
pFormatCtx = video_info(file1_info.FILE_NAME);
file1_info.FILE_SIZE = pFormatCtx->file_size;

//send video information
sprintf(buffer,"%s\t%s\t%d\t%d\t%d\n",file1_info.HOST_NAME,
file1_info.FILE_NAME,file1_info.FILE_SIZE,file1_info.BIT_RATE);
send(socket_master,buffer,BUFFER_SIZE,0);

//receive port & duration to use
recv(socket_master, buffer, BUFFER_SIZE, 0);
sscanf(buffer,"%d\t%d\n",&file1_info.PORT,&file1_info.DURATION);

//connect to server
int connection(char *HOST_NAME, char *PORT){

//create a master socket and check it
socket_master = socket(AF_INET,SOCK_STREAM,0);

//define server information
server.sin_family = AF_INET;                //source type (Internet)
server.sin_port = htons(atoi(PORT));       //source port (recived port)
hp = gethostbyname(HOST_NAME);
bcopy(hp -> h_addr, &server.sin_addr.s_addr, hp -> h_length); //source IP

//connect to server
connect(socket_master, (struct sockaddr *) &server, address_size);
return socket_master;}

//get video information
AVFormatContext *video_info(char *file_name){
av_open_input_file(&pFormatCtx, file_name, NULL, 0, NULL);
av_find_stream_info(pFormatCtx);
return pFormatCtx;}

```

الشكل (5-17) برنامج سي المسؤول عن اتصال المصدر مع المخدم.

عند ورود أي طلب من المستخدم إلى المصدر، فإنه يستقبل معلومات عن تخصيص أجزاء الملف ضمن حيزات الوصلة (VC). يبين الشكل (5-18) برنامج سي المكتوب من أجل التعامل مع طلبات المستخدمين.

```
//struct couple has a link and a content slot
typedef struct Couple{int link_slot, content_slot;}Couple;

//struct allocate link to content slot
typedef struct Slot_alloc{Couple couple[TL+1];}Slot_alloc;

///listening loop until receive client request

while(1){
//accept new connection
socket_new = accept(socket_master, (struct sockaddr *) &client,
&client_length));
int pid = fork();
if(pid < 0) printf("error2\n");
if(pid == 0){
close(socket_master);
do_stuff(socket_new, file1_info.FILE_NAME, TS, pFormatCtx);
exit(0);}
else close(socket_new);}

///this code is repeated for each client (socket)

void do_stuff(int socket_new, char *file_name, int TS, AVFormatContext
*pFormatCtx){
Slot_alloc VC;
recv(socket_new, &VC, sizeof(VC), 0);}
```

الشكل (18-5) برنامج سي المسؤول عن استقبال طلبات المستخدمين

في حالة مشاركة الملفات يقوم المصدر بتحديد المعلومات الأساسية للملف، وهي اسم الملف وحجمه، ومن ثم إرسالها إلى المستخدم، أما في حالة البث الفيديوي يحدد المصدر معلومات الترميز المستخدم في الفيديو وإرساله إلى المستخدم. تتضمن معلومات الترميز التي قمنا بإرسالها إلى المستخدم: اسم ترميز الفيديو والصوت، أبعاد شاشة العرض (الارتفاع $\times$ العرض width)، نمط البيكسلات pix_fmt، ومعدل البتات bit_rate، وحجم الإطار frame_size، وعدد أفقية الصوت channels، ومعدل العينات sample_rate. يوضح الشكل (19-5) كيفية استخلاص معلومات الترميز وإرسالها إلى المستخدم بلغة سي.

```
//codec structure
typedef struct codec_info{
    char vcodec[32], acodec[32];
    int height, width, channels, sample_rate;
    int bit_rate, frame_size;
    enum PixelFormat pix_fmt;
    int vstream, astream; AVRational sample_aspect_ratio;}codec_info;

void do_stuff(int socket_new, char *file_name, int TS, AVFormatContext
*pFormatCtx){
    codec_info codec1;
    codec1 = codec_info_get(pFormatCtx);
    send(socket_new, &codec1, sizeof(codec1), 0);
}

void open_decoder(AVStream *stream){
    AVCodec *Codec;
    Codec = avcodec_find_decoder(stream->codec->codec_id);
    if(Codec == NULL) fprintf(stderr,"Un supported codec !\n");
    if(avcodec_open(stream->codec,Codec) < 0)
    fprintf(stderr,"error during open decoder !\n");
}

codec_info codec_info_get(AVFormatContext *pFormatCtx){
    codec_info codec1;
    codec1.vstream=-1; codec1.astream=-1;
    for(i=0; i<pFormatCtx->nb_streams; i++){
        if(pFormatCtx->streams[i]->codec->codec_type==CODEC_TYPE_VIDEO)
        {codec1.vstream=i; vst=pFormatCtx->streams[i];}
        if(pFormatCtx->streams[i]->codec->codec_type==CODEC_TYPE_AUDIO)
        {codec1.astream=i; ast=pFormatCtx->streams[i];}
    }
    if(codec1.vstream!=-1){
        open_decoder(vst);
        strcpy(codec1.vcodec,vst->codec->codec->name);
        codec1.height    =vst->codec->height ;
        codec1.width     =vst->codec->width  ;
        codec1.pix_fmt    =vst->codec->pix_fmt ;
        codec1.bit_rate   =vst->codec->bit_rate;
        codec1.frame_size=vst->codec->frame_size;
        codec1.sample_aspect_ratio=vst->codec->sample_aspect_ratio;
    }
    if(codec1.astream!=-1){
        open_decoder(ast);
        strcpy(codec1.acodec,ast->codec->codec->name);
        codec1.channels=ast->codec->channels;
        codec1.sample_rate=ast->codec->sample_rate;
    }
    return codec1;}

```

الشكل (19-5) برنامج سي المستخدم لإيجاد معلومات الترميز وإرسالها إلى المستخدم

بعدها يقوم المصدر بقراءة بيانات الملف أو رزم الفيديو ووضعها ضمن حاوية بحجم ثابت مساوي لـ Bunit، يضاف إليها الرقم التسلسلي للحاوية ورقم الإطار الذي تنتمي إليه. يوضح الشكلان (5-20) و (5-21) برنامج سي الذي يقوم بإنشاء الحاوية ووضع رزم الفيديو ضمنها.

```
//struct container contains file
typedef struct container{
    int size[Bunit/2], sub_size[Bunit/2], stream[Bunit/2];
    uint8_t data[Bunit+1];
    int seq, ord;}container;
//link structure
typedef struct link_info{container Slot[TL+1]; int size;}link_info;

void do_stuff(int socket_new, char *file_name, int TS, AVFormatContext
*pFormatCtx){
extern link_info link1;
for(d=1;d<=file1_info.DURATION;d++){
ex1 = *buffer_fill(pFormatCtx, &ex1);
container_put(file_name,d,TS);
for(i=1;i<=TL;i++) {
if(VC.couple[i].content_slot>0)
link_slot_init(VC.couple[i].link_slot,VC.couple[i].content_slot);}
send(socket_new, &link1, link_size, 0); }}

void container_put(char *file_name,int d, int TS){
extern container con[TL];
for(i=1;i<=TS;i++){
for(j=1;j<=Bunit/2;j++){
con[i].size[j] = buf.size[j];
con[i].sub_size[j] = buf.sub_size[j];
con[i].stream[j] = buf.stream[j];
for(j=1;j<=Bunit;j++){con[i].data[j] = buf.buffer[j];}
con[i].seq = i;
con[i].ord = d;
}}}
```

الشكل (5-20) برنامج سي المسؤول عن تهيئة الحاوية.

```

ex *buffer_fill(AVFormatContext *pFormatCtx, ex *ex1){
if(ex1->extra>0){
if(ex1->extra<=Bunit){
    buf.stream[j] = ex1->index; buf.size[j] = ex1->size;
    buf.sub_size[j] = ex1->extra;
    for(k=1;k<=ex1->extra;k++) {buf.buffer[k]=*extra_buf; extra_buf++;i++;}
    total_size= total_size + ex1->extra; j++;}
if(ex1->extra>Bunit){buf.stream[j] = pkt.stream_index;
    buf.size[j] = ex1->size;
    buf.sub_size[j] = Bunit;
    for(k=1;k<=Bunit;k++) {buf.buffer[k]=*extra_buf; extra_buf++;}
    ex1->extra=ex1->extra-Bunit;
return ex1;}}

while((av_read_frame(pFormatCtx,&pkt))==0){
total_size= total_size + pkt.size;
if(Bunit - total_size >= 0){
    buf.stream[j]=pkt.stream_index; buf.size[j] = pkt.size;
    buf.sub_size[j] = pkt.size;
    while(size < pkt.size){
        buf.buffer[i]=*pkt.data;
        size = size + sizeof(uint8_t);
        pkt.data++; i++;}}

if(Bunit - total_size < 0){
    buf.stream[j]=pkt.stream_index; buf.size[j] = pkt.size; buf.sub_size[j] =
    Bunit - total_size + pkt.size;
    ex1->size=pkt.size; ex1->index=pkt.stream_index;
    for(k=1;k<=(Bunit - total_size + pkt.size);k++){
        buf.buffer[i]=*pkt.data;
        pkt.data++; i++;}
    extra_buf=pkt.data; ex1->extra=(total_size - Bunit);
    return ex1;} j++;}
return ex1;}

```

الشكل (21-5) برنامج سي المسؤول عن قراءة رزم الفيديو ووضعها ضمن الحاوية.

ومن ثمَّ يجري وضع كل حاوية بالمكان المخصص لها ضمن الوصلة وإرسالها إلى المستخدم، وهذا موضح بالشكل (22-5).

```
//link structure

typedef struct link_info{
container Slot[TL+1];
int size;}link_info;

void do_stuff(int socket_new, char *file_name, int TS, AVFormatContext
*pFormatCtx){
extern link_info link1;
for(i=1;i<=TL;i++) {
if(VC.couple[i].content_slot>0)
link_slot_init(VC.couple[i].link_slot,VC.couple[i].content_slot);}
send(socket_new, &link1, link_size, 0);
}}

void link_slot_init(int link_slot, int content_slot){
extern container con[TL];
extern link_info link1;
link1.Slot[link_slot] = con[content_slot];}
```

الشكل (5-22) برنامج سي المسؤول عن وضع الحاوية في المكان المخصص ضمن الوصلة.

4.7.5. التنفيذ البرمجي للمستخدم

يقوم المستخدم، في البداية، بإرسال اسم الملف المطلوب إلى المخدم، الذي يعيد إليه عناوين العقد التي يجب الاتصال بها مع أجزاء الملف المستقبلية من كل عقدة، وكذلك رقم المنفذ وحجم الملف في حالة مشاركة الملفات أو مدة الفيديو في حالة البث الفيديوي. يبين الشكل (5-23) برنامج سي المسؤول عن الاتصال بين المستخدم والمخدم.

```
//connect to server
socket_master = connection(argv[1],argv[2]);
//enter a video name and a host name
printf("Enter file name :\t"); scanf("%s", &file1.FILE_NAME);
printf("Enter host name :\t"); scanf("%s", &file1.HOST_NAME);
//send file name and host name
sprintf(buffer,"%s\t%s\n",file1.FILE_NAME,file1.HOST_NAME);
send(socket_master, buffer , BUFFER_SIZE, 0);
recv(socket_master,buffer,1,0);
strcpy(info,buffer);
if(strcmp(info,"y")!=0){printf("file not found !!!\n"); return -1;}
printf("file found !!!!\n");
//receive port & duration
recv(socket_master, buffer, sizeof(buffer) , 0);
sscanf(buffer,"%d\t%d\n",&file1.PORT,&file1.DURATION);
//receive source address
recv(socket_master,&allocate_table1,sizeof(allocate_table1) , 0);
```

الشكل (5-23) برنامج سي المسؤول عن الاتصال بين المستخدم والمخدم.

يتصل المستخدم بكل عنوان من العناوين المستقبلية بشكل متزامن، ويرسل لكل منها أجزاء الملف التي يريد استقبالها منه. بعدها يقوم المستخدم باستقبال وصلات الملف والحصول على الحاويات، ومن ثم إعادة ترتيبها تبعاً لرقمها التسلسلي. ثم تجري قراءة رزم الفيديو من ضمن الحاويات وتقسيمها إلى رزم فيديو ورزم صوت بحيث يجري التعامل مع كل منها بشكل مستقل، ومن ثم تحقيق التزامن فيما بينها. يبين الشكلان (5-24) و (5-25) برنامج سي المكتوب من أجل الاتصال بالمصدر.

```
//main function
while(1){
if(i<=allocate_table1.IP_NO[allocate_table1.HOST_ID]){
pthread_create(&scthread[i],NULL,source_connect,(void *)&scdata1[i]);
pthread_join(scthread[i],NULL); sleep(1);
pthread_create(&srthread[i],NULL,source_recv,(void *)&scdata1[i]);
sleep(1); i++;}
//connect to server
void source_connect(scdata *ptr){
scdata *data;
data = (scdata *)ptr;
socket_master = connection(data->hostname,data->port,"s");
send(socket_master, &data->vc, sizeof(data->vc), 0);}
```

الشكل (5-24) برنامج سي المكتوب من أجل الاتصال بالمصدر.

```

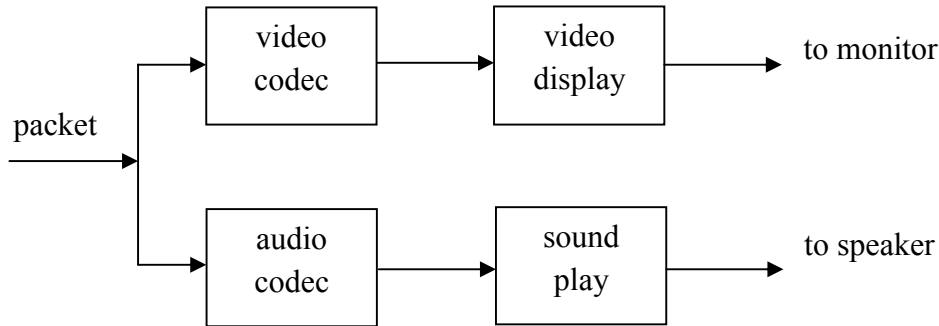
//recv slot allocation
void source_recv(sdata *ptr){
sdata *data;
data = (sdata *)ptr;
recv(srdata1[data->i].socket_master,&link1,srdata1[data->i].link_size,0);
for(i=1;i<=TL;i++){
    if(data->vc.couple[i].content_slot){
        con[link1.Slot[i].seq]=link1.Slot[i];
        cdata1.con[link1.Slot[i].seq]=con[link1.Slot[i].seq];
        cdata1.index=pkt_fill(&cdata1);
    }}

//get packet from container
int pkt_fill(cdata *cdata1){
while(cdata1->con[cdata1->j].size[j] > 0){
    pkt.size = cdata1->con[cdata1->j].size[j];
    pkt.stream_index = cdata1->con[cdata1->j].stream[j];
    if(cdata1->con[cdata1->j].sub_size[j]==cdata1->con[cdata1->j].size[j]){
        for(i=1;i<=cdata1->con[cdata1->j].sub_size[j];i++)
            {data[i]=cdata1->con[cdata1->j].data[k];k++;index=0;}}
    if(cdata1->con[cdata1->j].sub_size[j]!=cdata1->con[cdata1->j].size[j]){
        if((index+cdata1->con[cdata1->j].sub_size[j]< cdata1->con[cdata1->j].
            size[j]){
            if(index==0){
                for(i=index+1;i<=cdata1->con[cdata1->j].sub_size[j];i++)
                    {tmp_data[i]=cdata1->con[cdata1->j].data[k];k++;}
                extra_buf = &tmp_data[1];
                index=i-1;}
            else{
                for(i=1;i<=index;i++){tmp_data[i]=*extra_buf; extra_buf++;}
                for(i=index+1;i<=(cdata1->con[cdata1->j].sub_size[j]+index);i++)
                    {tmp_data[i]=cdata1->con[cdata1->j].data[k];k++;}
                extra_buf = &tmp_data[1];
                index=i-1;}}
        else{
            for(i=1;i<=index;i++){data[i]=*extra_buf; extra_buf++;}
            for(i=index+1;i<=cdata1->con[cdata1->j].size[j];i++)
                {data[i]= cdata1->con[cdata1->j].data[k];k++;}
            index=0;}}
    if(index==0){pkt.data=&data[1];
    if(pkt.stream_index==srdata1[cdata1->j].vstream)
        {video_display(pkt,cdata1->j);}
    if(pkt.stream_index==srdata1[cdata1->j].astream)
        {audio_display(pkt,cdata1->j);}
    }
    j++;}
return index;}

```

الشكل (25-5) برنامج سي المكتوب من أجل استقبال البيانات من المصدر.

بعد الحصول على رزم الفيديو، قمنا بفك ترميز الفيديو. ثم تهيئة الصورة وعرضها على الشاشة وذلك بعد الحصول على كامل الإطار. أما بالنسبة لرزم الصوت فقمنا بفك ترميزها، ثم وضعها ضمن رتل queue بحيث يجري استخدامها في الوقت المناسب بما يحقق التزامن بين الصوت والصورة. وتبين الأشكال (26-5) و (27-5) و (28-5) آلية التعامل مع الصوت والصورة، وبرنامج سي للتعامل مع الفيديو.



الشكل (26-5) آلية التعامل مع الفيديو.

```

//video display on screen
void video_display(AVPacket pkt, int j){
    avcodec_decode_video(srdata1[j].vCodecCtx,srdata1[j].vFrame,
    &frameFinished,pkt.data, pkt.size);
    if(frameFinished){
        SDL_LockYUVOverlay(srdata1[j].bmp);
        pict.data[0] = srdata1[j].bmp->pixels[0];
        pict.data[1] = srdata1[j].bmp->pixels[2];
        pict.data[2] = srdata1[j].bmp->pixels[1];
        pict.linesize[0] = srdata1[j].bmp->pitches[0];
        pict.linesize[1] = srdata1[j].bmp->pitches[2];
        pict.linesize[2] = srdata1[j].bmp->pitches[1];
        img_convert(&pict, PIX_FMT_YUV420P, (AVPicture *)srdata1[j].vFrame,
        srdata1[j].vCodecCtx->pix_fmt, srdata1[j].vCodecCtx->width,
        srdata1[j].vCodecCtx->height);
        SDL_UnlockYUVOverlay(srdata1[j].bmp);
        rect.x = 0;
        rect.y = 0;
        rect.w = srdata1[j].vCodecCtx->width;
        rect.h = srdata1[j].vCodecCtx->height;
        SDL_DisplayYUVOverlay(srdata1[j].bmp, &rect);
    }
}
  
```

الشكل (27-5) برنامج سي لتهيئة الشاشة من أجل عرض الفيديو

```
//play sound
int audio_decode_frame(uint8_t *audio_buf, int buf_size){
for(;;) {
while(is->audio_pkt_size > 0){
data_size = buf_size;
len1 = avcodec_decode_audio2(is->aCodecCtx, (int16_t *)audio_buf,
&data_size, is->audio_pkt_data, is->audio_pkt_size);
if(len1 < 0){/* if error, skip frame */is->audio_pkt_size = 0; break;}
is->audio_pkt_data += len1; is->audio_pkt_size -= len1;
if(data_size <= 0) {/* No data yet, get more frames */      continue;}
/* We have data, return it and come back for more later */
return data_size;}
/* next packet */
if(packet_queue_get(&is->audioq, pkt, 1) < 0) return -1;
is->audio_pkt_data = pkt->data; is->audio_pkt_size = pkt->size;}}

void audio_callback(UInt8 *stream, int len){
while(len > 0) {
if(is->audio_buf_index >= is->audio_buf_size) {
/* We have already sent all our data; get more */
audio_size = audio_decode_frame(is->audio_buf, sizeof(is->audio_buf));
if(audio_size < 0){/* If error, output silence */is->audio_buf_size =
1024;memset(is->audio_buf, 0, is->audio_buf_size);}
else is->audio_buf_size = audio_size;
is->audio_buf_index = 0;}
len1 = is->audio_buf_size - is->audio_buf_index;
if(len1 > len) len1 = len;
memcpy(stream, (uint8_t *)is->audio_buf + is->audio_buf_index, len1);
len -= len1; stream += len1; is->audio_buf_index += len1;}}
```

الشكل (28-5) برنامج سي المكتوب من أجل عرض الفيديو على الشاشة

بالوقت نفسه يقوم المستخدم بتهيئة الاتصال والعمل بحالة استماع بحيث يتلقى طلبات مستخدمين آخرين وعندها يتعامل مع هذه الطلبات وكأنه مصدر للملف. يبين الشكل (29-5) برنامج سي من أجل إعادة الإرسال.

```
void re_send(int socket_new, char *file_name, int TS, int DURATION){
    recv(socket_new, &VC, sizeof(VC), 0);
    link_size = (TS+2)*sizeof(container);
    send(socket_new, &link_size, sizeof(link_size),0);
    for(d=1;d<=DURATION;d++){
        container_put(TS,file_name,m,d);
        for(i=1;i<=TL;i++){
            if(VC.couple[i].content_slot>=0) link_slot_init(&link1,
            VC.couple[i].link_slot,VC.couple[i].content_slot);
        }
        ss = send(socket_new, &link1, link_size, 0);
    }
}
```

الشكل (5-29) برنامج سي من أجل إعادة الإرسال.

8.5. الخلاصة

عرضنا في هذا الفصل تصميم تطبيق ند لند من أجل مشاركة الملفات وبث الفيديو. حيث قمنا بشرح البنية المقترحة للتطبيق، والتي تتألف من ثلاثة مكونات أساسية؛ وهي المخدم المركزي المسؤول عن إدارة وتنظيم العقد ضمن الشبكة، والمصدر الذي يمتلك ملف أو أكثر ويرغب بمشاركتها عبر الشبكة، وأخيراً المستخدم.

كما تضمن الفصل توضيحاً للآلية المعتمدة من أجل تنظيم العقد ضمن الشبكة، وكذلك إعادة بناء الشبكة في حال فشل بعض العقد أو مغادرتها الشبكة بشكل مفاجئ.

كما اشتمل الفصل توضيح للبرامج المكتوبة بلغة سي من أجل توصيف عمل التطبيق وآلية اتصال مكوناته مع بعضها البعض إضافةً إلى المخططات التدفقية ذات الصلة.

النتائج

سنوضح في هذا الفصل نتائج اختبار التطبيق المقترح وتأثير تغيير مستويات التطبيق على أدائه. تحققنا من أداء التطبيق بحالتين وهما: الفتح بعد الإتيان بالملف ، وفيها يجري تخزين الملف بالكامل عند المستخدم قبل استخدامه، والثانية هي تشغيل الملف أثناء الإتيان به، وذلك بواسطة محاكاة التطبيق.

1.6. إعداد بيئة الاختبار

قمنا باختبار التطبيق باستخدام حاسب محمول من نوع Toshiba L40، له المواصفات الآتية:

- Intel Pentium Dual CPU T2370
- Total Usable Physical Memory 1014MB
- Realtek RTL8139/810x Family Fast Ethernet NIC
- Realtek RTL8187B Wireless 802.11g 54Mbps USB 2.0 Network Adapter
- Mandriva Linux Operating System

استخدمنا نظام التشغيل لينوكس Linux من أجل بناء شبكة افتراضية، تتألف من مخدم مركزي، ومصدر واحد، وعدد من المستخدمين.

يتيح نظام لينوكس العديد من الميزات للتعامل مع الشبكات، ومنها مشاركة موارد الجهاز لبناء عدة مستخدمين، حيث يُمثل كل مستخدم بواجهة تكافئ عقدة بالشبكة، ويجري إعطاءها عنوان شبكة خاص ويجري التعامل معها وكأنها مستخدم مستقل ضمن الشبكة. قمنا بإنشاء خمس واجهات وإعطائها عناوين الشبكة الآتية:

المخدم المركزي	: 192.168.1.5
المصدر	: 192.168.1.6
المستخدم الأول	: 192.168.1.10
المستخدم الثاني	: 192.168.1.11
المستخدم الثالث	: 192.168.1.12

ثم قمنا بوضع البرنامج المخصص في كل من واجهة المخدم المركزي والمصدر والمستخدمين، من ثمَّ التحقق من أداء التطبيق.

قمنا بتنفيذ التطبيق بشكل افتراضي وذلك لعدم توافر مخابر مجهزة بنظام تشغيل لينوكس ضمن القسم، وتعطل الشبكة في مخبر اللينوكس الوحيد المتوافر في قسم الحواسيب والأتمتة. كما أننا لم نتمكن من الحصول على دعم من البحث العلمي من أجل شراء عدة حواسيب كما ورد في استمارة البحث بهدف تحقيق شبكة فعلية.

2.6. المتوسطات الأساسية المستخدمة في اختبار التطبيق

استخدمنا من أجل اختبار التطبيق مجموعة من المتوسطات الموضحة بالجدول (1-6).

الجدول (1-6) المتوسطات المستخدمة في اختبار التطبيق.

الموسم	الرمز	التعريف
عرض مجال الوصلة	BWL	عرض المجال الكلي المتاح لوصلة العقدة
وحدة عرض المجال	Bunit	قيمة افتراضية لوحدة عرض المجال من أجل تبسيط آلية عمل التطبيق
عرض مجال الوصلة الصاعدة	BWu	عرض المجال المتاح من قبل المستخدم لإرسال الملف إلى بقية المستخدمين
عرض مجال الوصلة الهابطة	BWd	عرض المجال المستخدم لاستقبال الملف
عدد حيزات الوصلة	TL	عدد يمثل تقسيم عرض مجال الوصلة على وحدة عرض المجال $TL=BWL/Bunit$
عدد حيزات المحتوى	TS	عدد يمثل تقسيم حجم الملف على وحدة عرض المجال $TS=Size/Bunit$

3.6. مقارنة التطبيق مع بنية المخدم/زبون

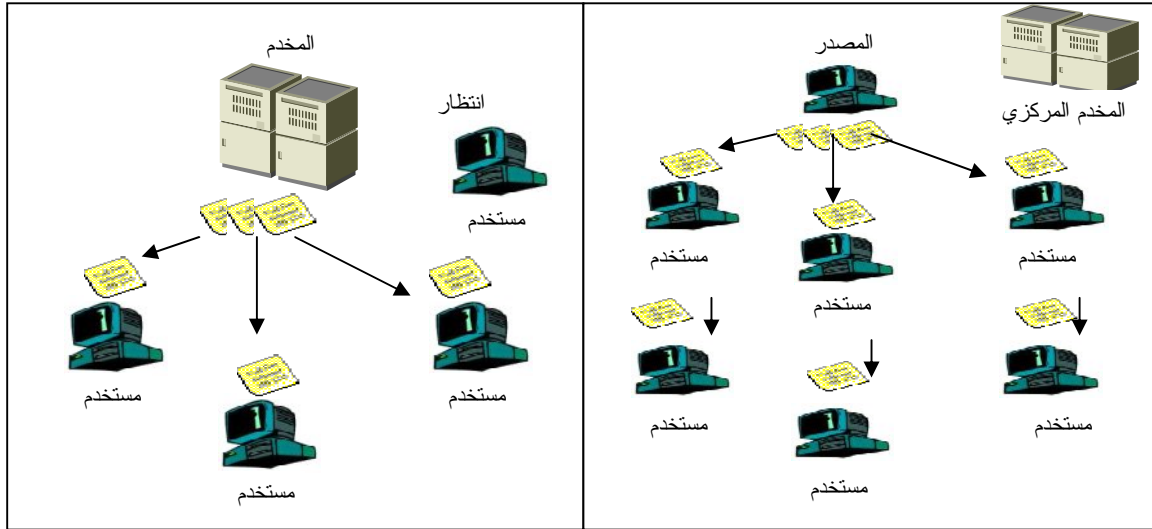
قمنا بمقارنة أداء التطبيق المقترح مع بنية المخدم/زبون، التي تعتمد على وجود مخدم مسؤول عن إجراء كامل الاتصالات بالشبكة، من حيث زيادة عدد المستخدمين للتطبيق، وكذلك زمن استقبال الملف. وهذه النتائج مبينة بالجدولين (2-6) و (3-6) على التوالي. ويبيّن الشكلان (1-6) و (2-6) مخطط الاتصال بين العقد لتطبيق المخدم/زبون وتطبيق الند للند المقترح، وذلك بناءً على النتائج المتضمنة في الجدولين.

الجدول (2-6) مقارنة تطبيق الند للند المقترح مع بيئة المخدم/زبون من حيث عدد المستخدمين

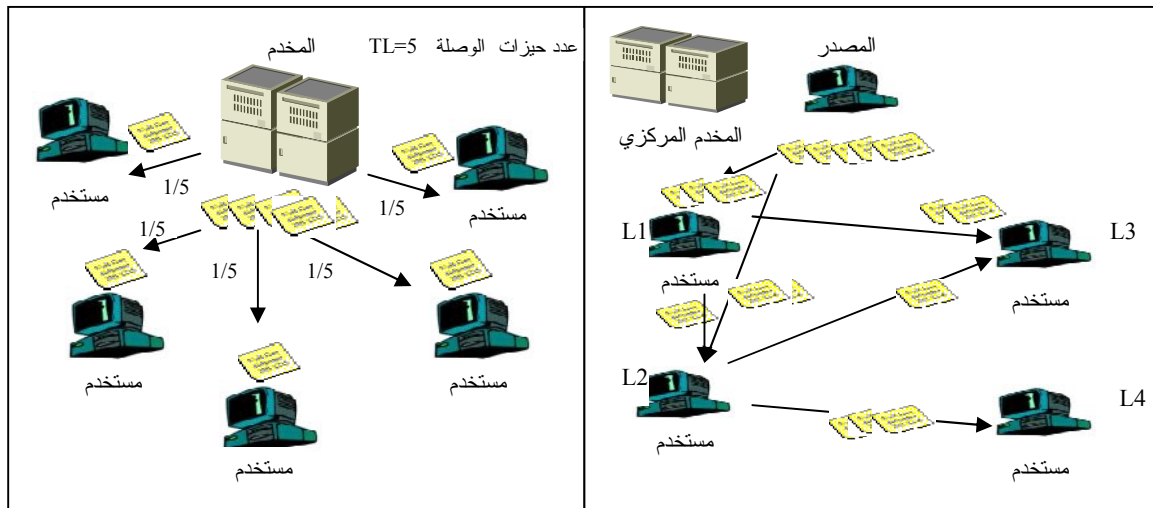
Our P2P application	C\S	TS=1	
> 6	3		TL=3
> 8	4		TL=4
> 10	5		TL=5

الجدول (3-6) مقارنة تطبيق الند للند المقترح مع بيئة المخدم/زبون من حيث زمن الاستقبال للملف.

Our P2P application				C\S	No. clients =5	File size
L4	L3	L2	L1			
23	20	18	14	28		3.2 KB
78	70	65	50	100		15.7 KB



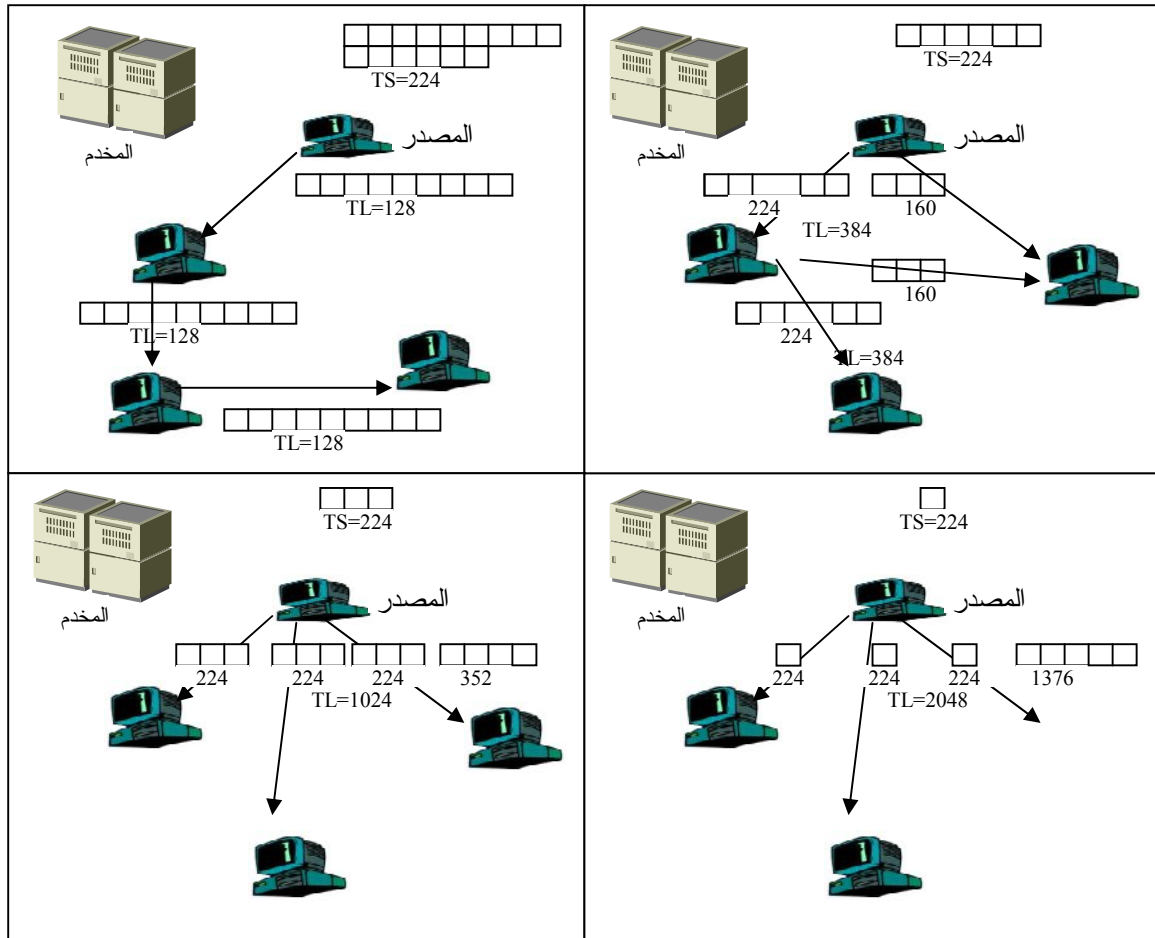
الشكل (1-6) مقارنة عدد مستخدمي تطبيق المخدم/زبون وتطبيق الند للند المقترح من أجل TL=3.



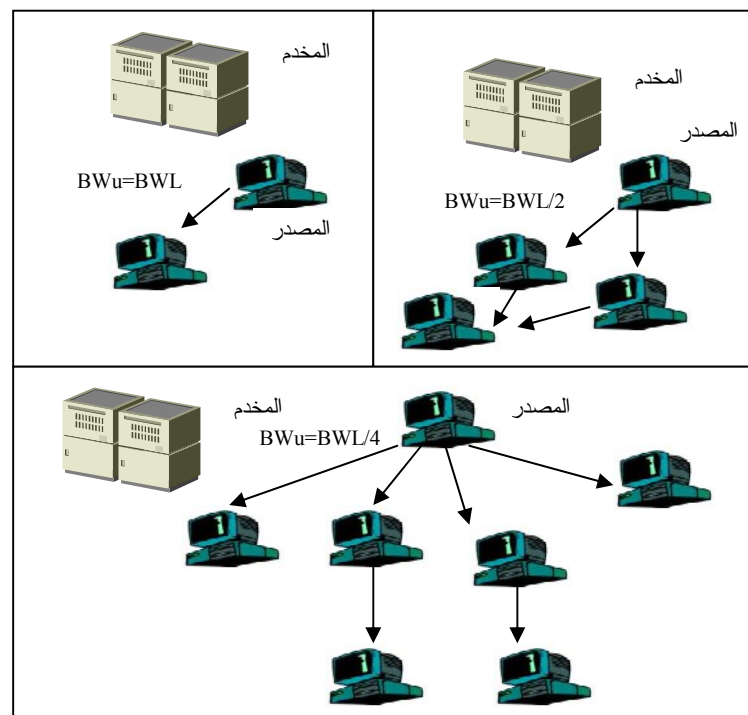
الشكل (2-6) مخطط اتصال العقد بحالة المخدم/زبون وتطبيق الند للند المقترح.

4.6. أداء التطبيق من أجل مشاركة الملفات

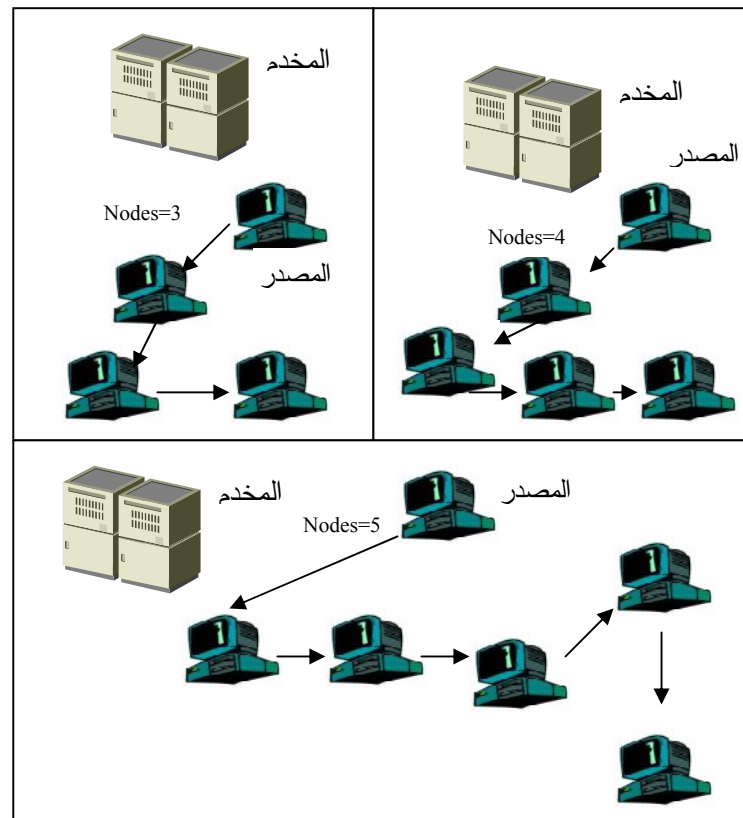
نبيّن فيما يأتي نتائج اختبار التطبيق من أجل مشاركة الملفات، وفيها يجري الإتيان بكامل الملف وتخزينه، ومن ثمّ استخدامه. أجرينا عدة اختبارات من أجل التحقق من أداء التطبيق في حالات مختلفة، حيث قمنا بقياس الزمن اللازم للاستقبال وذلك مع تغيير بعض مستويات التطبيق ومنها: عرض مجال الوصلة BWL، وعرض مجال الوصلة الهابطة BWd، وعدد المستخدمين، وحجم الملفات. من أجل قيم مختلفة لعرض مجال الوصلة الكلي، وعرض مجال وصلة صاعدة وهابطة مختلفان، و عدد مختلف للعقد، تبيّن الأشكال (3-6) و(4-6) و(5-6) مخطط الشبكة من أجل الحالات الثلاثة المدروسة.



الشكل (3-6) مخطط الشبكة من أجل عرض مجال وصلة 2Mbps، 1Mbps، 384Kbps، 128Kbps، على التوالي. ومن أجل حجم ملف 28KB (224Kb).



الشكل (4-6) مخطط الشبكة من أجل قيم عرض مجال وصلة هابطة مختلفة بالنسبة لعرض مجال الوصلة الكلي.

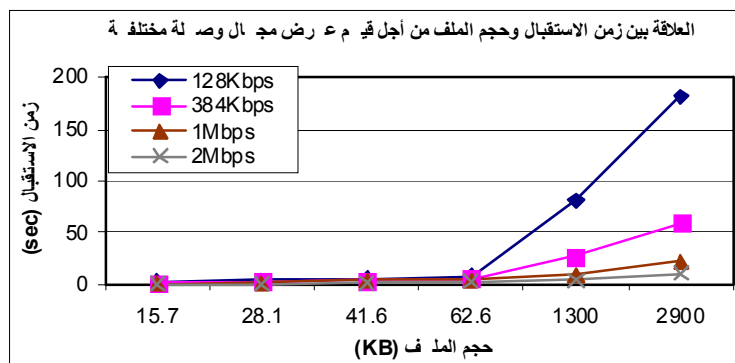


الشكل (5-6) مخطط الشبكة من أجل عدد عقد مختلف.

1.4.6. أداء التطبيق من أجل أنواع اتصال مختلفة

قمنا في هذه المرحلة، بنقل ملفات بأنواع وأحجام مختلفة وقياس الزمن اللازم للاستقبال، وذلك من أجل أنواع اتصال وعرض مجال مختلف.

يبين الشكل (6-6) العلاقة بين زمن الاستقبال وحجم الملف من أجل قيم مختلفة لعرض مجال الوصلة BWL.



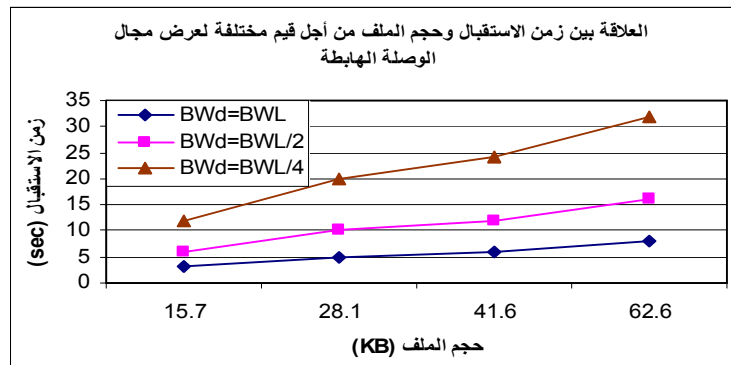
الشكل (6-6) علاقة حجم الملف بزمن الاستقبال من أجل قيم مختلفة لعرض مجال الوصلة BWL.

يلاحظ من الشكل تناقص الزمن اللازم لاستقبال الملف مع ازدياد عرض مجال الوصلة المستخدمة، فمن أجل حجم ملف 1300KB، كان زمن الاستقبال بعرض مجال 128Kbps مساوي لـ 81 ثانية، بينما يساوي 5 ثانية عند عرض مجال وصلة 2Mbps.

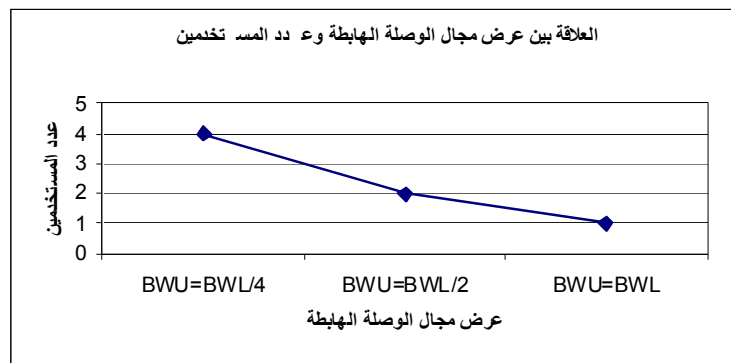
2.4.6. دراسة مستويات التطبيق من أجل عرض مجال وصلة 128Kbps

قمنا في هذه المرحلة، بنقل ملفات بأنواع وأحجام مختلفة وقياس الزمن اللازم للاستقبال، وذلك من أجل عرض مجال وصلة $BWL=128Kbps$.

يبين الشكل (6-7) العلاقة بين زمن الاستقبال وحجم الملف من أجل قيم مختلفة لعرض مجال الوصلة الهابطة بالمقارنة مع العرض المجال الكلي للوصلة. وتمثل الحالة الثانية عندما $BWd=BWL$ تناظر عرض المجال للوصلتين الهابطة والصاعدة. في حين يبين الشكل (6-8) العلاقة بين عدد المستخدمين وعرض المجال المختار للوصلة الهابطة.



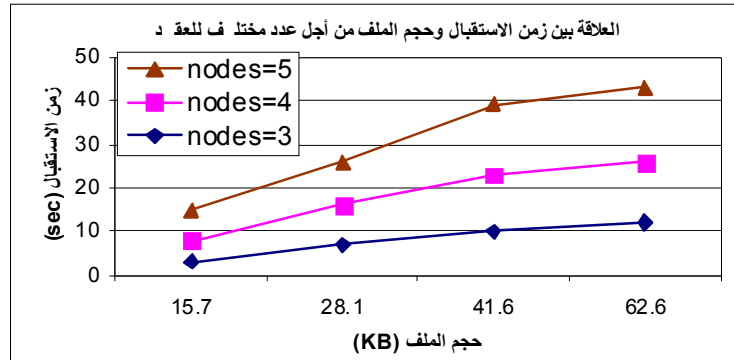
الشكل (6-7) العلاقة بين زمن الاستقبال وحجم الملفات من أجل عرض مجال وصلة 128Kbps.



الشكل (6-8) العلاقة بين عدد المستخدمين وعرض مجال الوصلة الهابطة.

يلاحظ من مقارنة الشكلان (6-7) و (6-8)، ازدياد الزمن المطلوب لاستقبال الملفات عند انقاص عرض مجال الوصلة الهابطة، لكن ذلك ينعكس إيجاباً على عدد المستخدمين الممكن تخدمه بالتطبيق.

قمنا في هذه الخطوة بزيادة عدد المستخدمين ودراسة تأثير ذلك على زمن الاستقبال للملف. يبيّن الشكل (6-9) العلاقة بين زمن الاستقبال وحجم الملف من أجل قيم مختلفة لعدد المستخدمين.

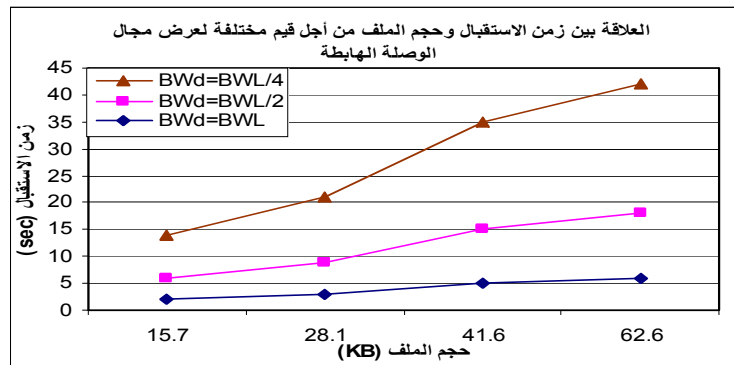


الشكل (6-9) العلاقة بين زمن الاستقبال وحجم الملف من أجل عدد مختلف للعقد وعند عرض مجال وصلة يساوي 128Kbps.

من الواضح أن زيادة عدد المستخدمين يزيد الزمن المطلوب لاستقبال الملفات، وذلك لوجود وقت بين استقبال الملف من قبل المستخدم، ومن ثم إعادة إرساله للمستخدم التالي. ولكن هذا التأخير ذو قيمة مقبولة حيث لا تتجاوز 3 ثوانٍ.

3.4.6. دراسة مستويات التطبيق من أجل عرض مجال وصلة 384Kbps

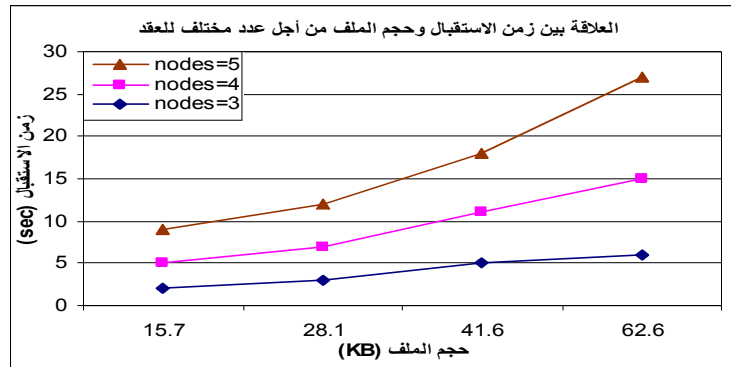
قمنا في هذه المرحلة، بتكرار الخطوات السابقة، وذلك من أجل عرض مجال وصلة $BWL=384Kbps$. يبيّن الشكل (6-10) العلاقة بين زمن الاستقبال وحجم الملف من أجل قيم مختلفة لعرض مجال الوصلة الهابطة BWd بالمقارنة مع عرض مجال الوصلة الكلي BWL .



الشكل (6-10) العلاقة بين زمن الاستقبال وحجم الملفات من أجل عرض مجال وصلة 384Kbps.

يلاحظ من الشكل ازدياد الزمن اللازم لاستقبال الملف مع انقاص عرض مجال الوصلة الهابطة، ولكن ذلك ينقص من عرض المجال المتاح للوصلة الصاعدة، ومن ثم يقلل من عدد المستخدمين للتطبيق.

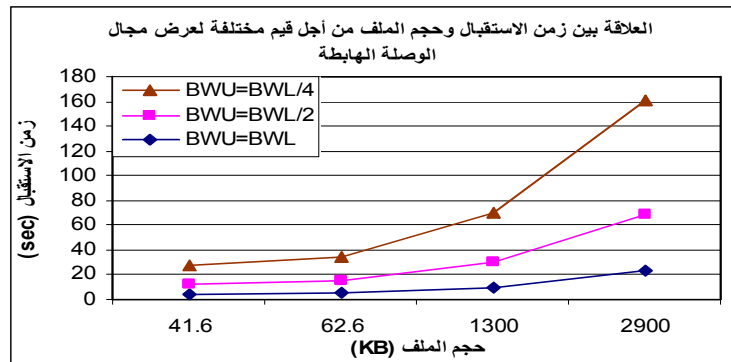
يبين الشكل (6-11) العلاقة بين زمن الاستقبال وحجم الملف من أجل قيم مختلفة لعدد المستخدمين.



الشكل (6-11) العلاقة بين زمن الاستقبال وحجم الملف من أجل عدد مختلف للعقد وعند عرض مجال وصلة يساوي 384Kbps. يؤدي ازدياد عدد المستخدمين للتطبيق إلى زيادة الزمن اللازم لاستقبال الملف، وذلك نتيجة وجود تأخير زمني بين استقبال المستخدم للملف وإعادة إرساله للمستخدمين الآخرين.

4.4.6. دراسة مستويات التطبيق من أجل عرض مجال وصلة 1Mbps

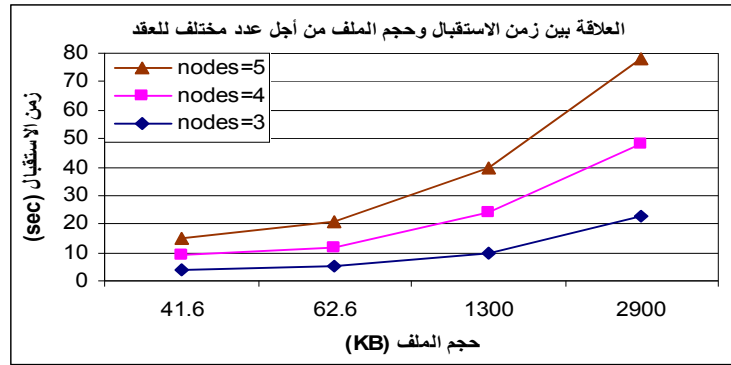
قمنا في هذه المرحلة، بتكرار الخطوات السابقة، وذلك من أجل عرض مجال وصلة $BWL=1Mbps$. يبين الشكل (6-12) العلاقة بين زمن الاستقبال وحجم الملف من أجل قيم مختلفة لعرض مجال الوصلة الهابطة بالمقارنة مع عرض مجال الوصلة.



الشكل (6-12) العلاقة بين زمن الاستقبال وحجم الملفات من أجل عرض مجال وصلة 1Mbps.

من الملاحظ تناقص الزمن اللازم لاستقبال الملفات مع ازدياد عرض مجال الوصلة، حيث استطعنا استقبال ملفات بحجم أكبر وبزمن أقل بالمقارنة مع عرض مجال وصلة 128Kbps و 384Kbps.

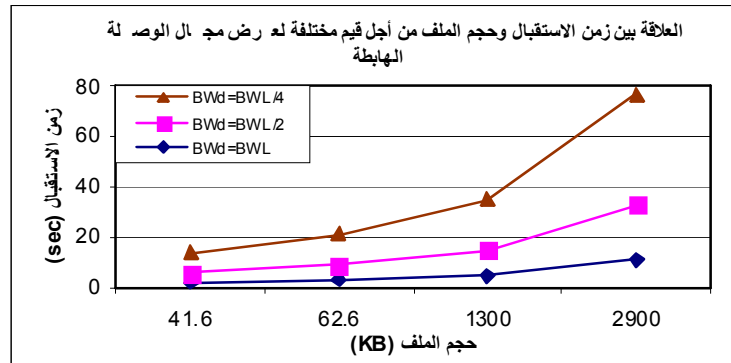
يبين الشكل (6-13) العلاقة بين زمن الاستقبال وحجم الملف من أجل قيم مختلفة لعدد المستخدمين.



الشكل (6-13) العلاقة بين زمن الاستقبال وحجم الملف من أجل عدد مختلف للعقد وعند عرض مجال وصلة يساوي 1Mbps. من الواضح أن زيادة عدد المستخدمين يزيد الزمن المطلوب لاستقبال الملفات، ولكن هذا التأخير ذو قيمة مقبولة لا تتجاوز 5 ثوانٍ.

5.4.6. دراسة مستويات التطبيق من أجل عرض مجال وصلة 2Mbps

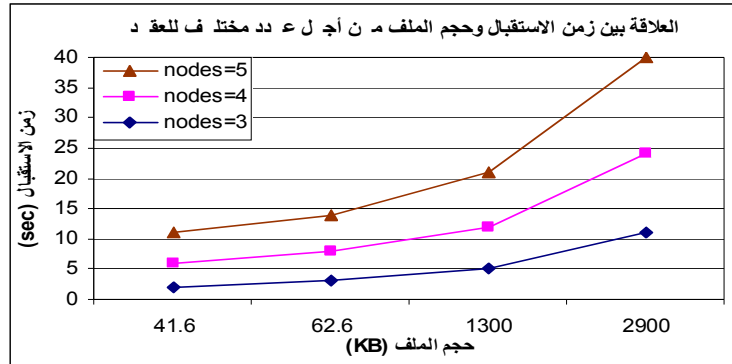
قمنا في هذه المرحلة، بتكرار الخطوات السابقة، وذلك من أجل عرض مجال وصلة $BWL=2Mbps$. يبين الشكل (6-14) العلاقة بين زمن الاستقبال وحجم الملف من أجل قيم مختلفة لعرض مجال الوصلة المقارنة مع عرض مجال الوصلة.



الشكل (6-14) العلاقة بين زمن الاستقبال وحجم الملفات من أجل عرض مجال وصلة 2Mbps.

نلاحظ من الشكل أنه من أجل عرض مجال وصلة 2Mbps حصلنا على أزمنة استقبال لا تتجاوز 80 ثانية من أجل حجم ملفات يصل حتى 3MB تقريباً.

يبين الشكل (6-15) العلاقة بين زمن الاستقبال وحجم الملف من أجل قيم مختلفة لعدد المستخدمين.



الشكل (6-15) العلاقة بين زمن الاستقبال وحجم الملف من أجل عدد مختلف للعقد وعند عرض مجال وصلة يساوي 2Mbps.

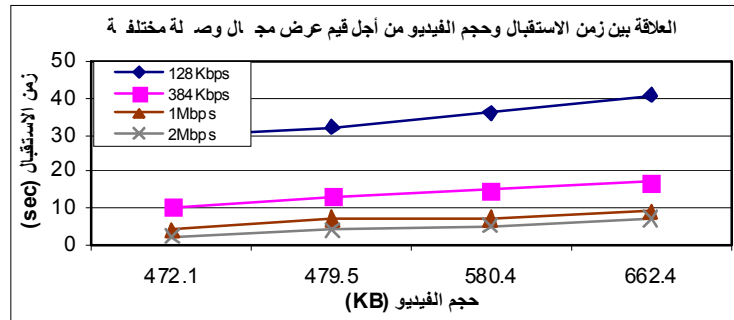
يوضح الشكل التصعد المرتفع للتطبيق، حيث من الممكن زيادة عدد المستخدمين بدون زيادة ملحوظة في زمن الاستقبال.

5.6. اختبار أداء التطبيق من أجل البث الفيديوي

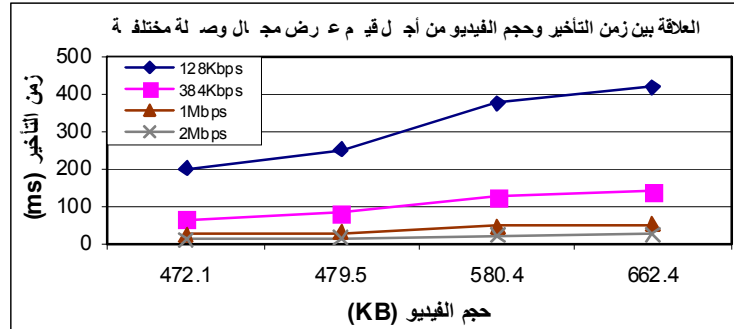
أجرينا مجموعة من التعديلات بحيث يتناسب التطبيق مع بث الفيديو، وفيه يجري بدء عرض الفيديو أثناء الإتيان به، بعد تأخير زمني بسيط. ثم أجرينا عدة اختبارات للتحقق من أداء التطبيق باستخدام المحاكاة، بما في ذلك الزمن اللازم لاستقبال الفيديو وزمن التأخير play back اللازم لتخزين ومعالجة الفيديو قبل بدء عرضه. كما قمنا بالتحكم بمجموعة من المتوسطات لدراسة تأثير كل منها على عمل التطبيق وأدائه، ومنها: عرض المجال الكلي للوصلة BWL، وعلاقته مع عرض مجال الوصلة الهابطة BWd، وعدد مستخدمي التطبيق.

1.5.6. أداء التطبيق من أجل أنواع اتصال مختلفة

قمنا في هذه المرحلة، بنقل ملفات فيديو بأحجام مختلفة وقياس الزمن اللازم للاستقبال وكذلك زمن التأخير قبل بدء عرض الفيديو، وذلك من أجل أنواع اتصال وعرض مجال مختلف. يبين الشكل (6-16) العلاقة بين زمن الاستقبال وحجم الملف من أجل قيم مختلفة لعرض مجال الوصلة BWL. أما الشكل (6-17) فيبين علاقة زمن التأخير مع حجم الملف وذلك من أجل قيم مختلفة لعرض مجال الوصلة.



الشكل (16-6) علاقة حجم الفيديو بزمان الاستقبال من أجل قيم مختلفة لعرض مجال الوصلة BWL.

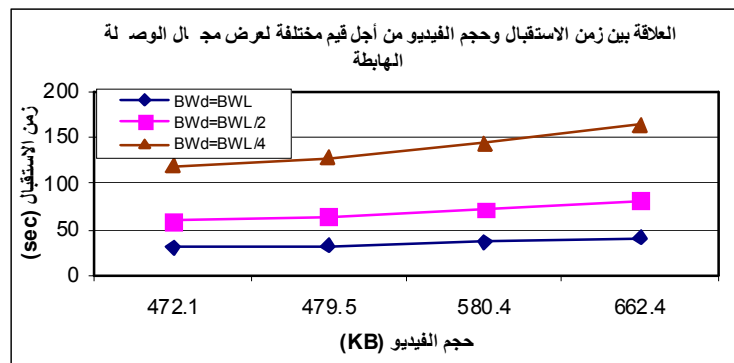


الشكل (17-6) علاقة حجم الفيديو بزمان التأخير من أجل قيم مختلفة لعرض مجال الوصلة BWL.

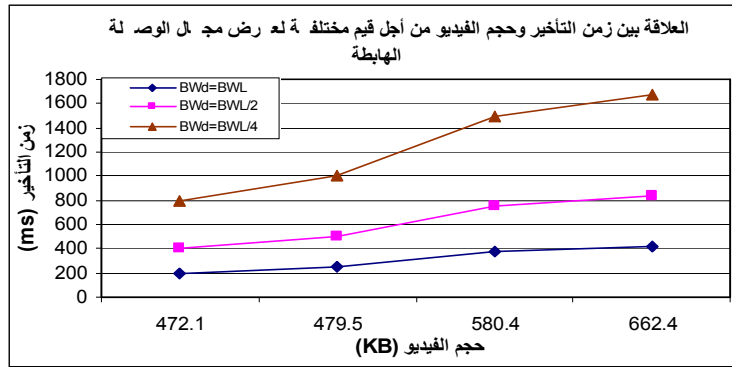
يلاحظ من الشكلين السابقين أن زيادة عرض مجال الوصلة BWL تؤدي إلى إنقاص الزمن المطلوب لاستقبال الفيديو وزمن التأخير اللازم لبدء عرض الفيديو على الشاشة.

2.5.6. دراسة موسطات التطبيق من أجل عرض مجال وصلة 128Kbps

قمنا في هذه المرحلة، بنقل ملفات فيديو مختلفة وقياس الزمن اللازم للاستقبال، وذلك من أجل عرض مجال وصلة $BWL=128Kbps$. يبين الشكل (6-18) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل قيم مختلفة لعرض مجال الوصلة الهابطة بالمقارنة مع عرض مجال الوصلة. في حين يبين الشكل (6-19) العلاقة بين زمن التأخير وحجم الفيديو من أجل قيم مختلفة لعرض مجال الوصلة الهابطة بالمقارنة مع عرض مجال الوصلة.



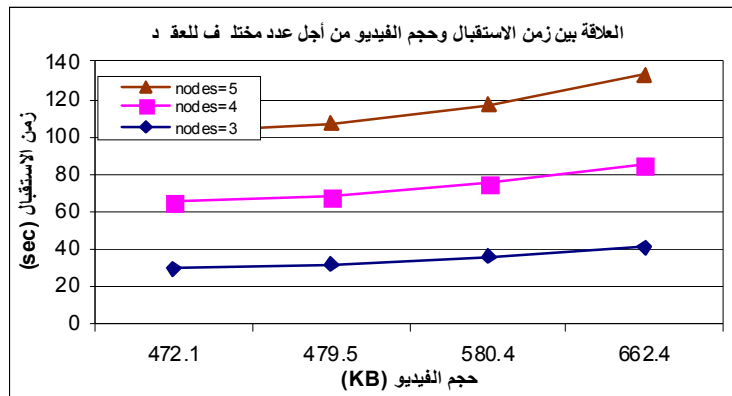
الشكل (18-6) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل عرض مجال وصلة 128Kbps.



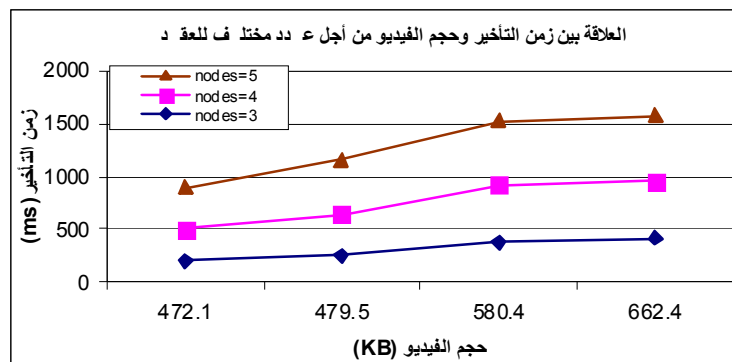
الشكل (6-19) العلاقة بين زمن التأخير وحجم الفيديو من أجل عرض مجال وصلة 128Kbps.

من الملاحظ ازدياد زمن الاستقبال وزمن التأخير مع إنقاص عرض مجال الوصلة الهابطة المتاحة، ولكن ذلك يؤدي لازدياد عرض مجال الوصلة الصاعدة المتاحة ومن ثمَّ ازدياد عدد المستخدمين للتطبيق.

قمنا في هذه الخطوة بزيادة عدد المستخدمين ودراسة تأثير ذلك على زمن الاستقبال وزمن التأخير للفيديو. يبيّن الشكلان (6-20) و (6-21) العلاقة بين زمن الاستقبال وزمن التأخير مع حجم الفيديو من أجل قيم مختلفة لعدد المستخدمين.



الشكل (6-20) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل عدد مختلف للعقد وعند عرض مجال وصلة يساوي 128Kbps.

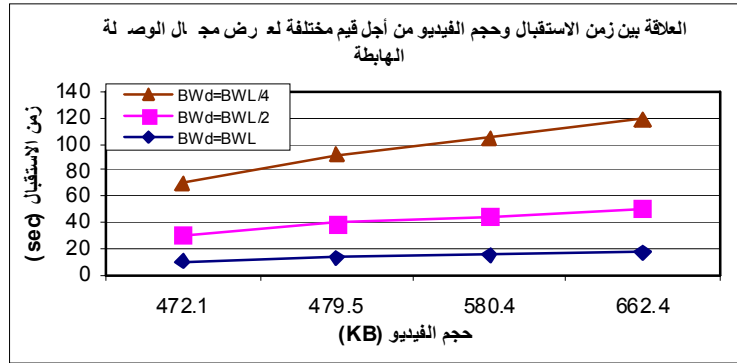


الشكل (6-21) العلاقة بين زمن التأخير وحجم الفيديو من أجل عدد مختلف للعقد وعند عرض مجال وصلة يساوي 128Kbps.

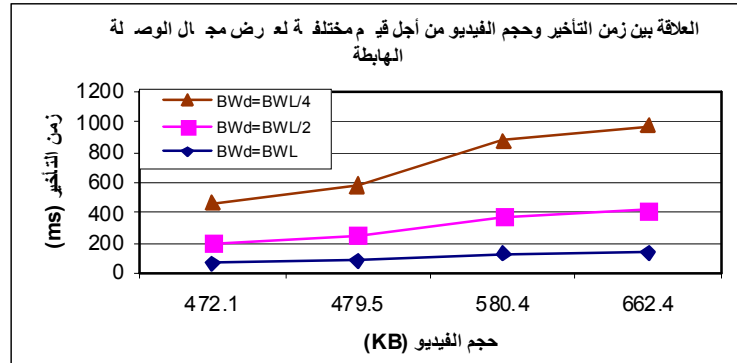
يؤدي ازدياد عدد مستخدمي التطبيق لازدياد الزمن اللازم لاستقبال الفيديو وكذلك زمن التأخير قبل بدء عرضه بقيم مقبولة. ولذلك يُعدّ التطبيق ذو تصعد مرتفع، حيث يمكن زيادة عدد المستخدمين دون تأثير ملحوظ على أداء التطبيق.

3.5.6. دراسة مستويات التطبيق من أجل عرض مجال وصلة 384Kbps

قمنا في هذه المرحلة، بتكرار الخطوات السابقة، وذلك من أجل عرض مجال وصلة $BWL=384Kbps$. يبيّن الشكل (6-22) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل قيم مختلفة لـ BWd مع عرض مجال وصلة 384Kbps. ويبيّن الشكل (6-23) العلاقة بين زمن التأخير وحجم الفيديو.



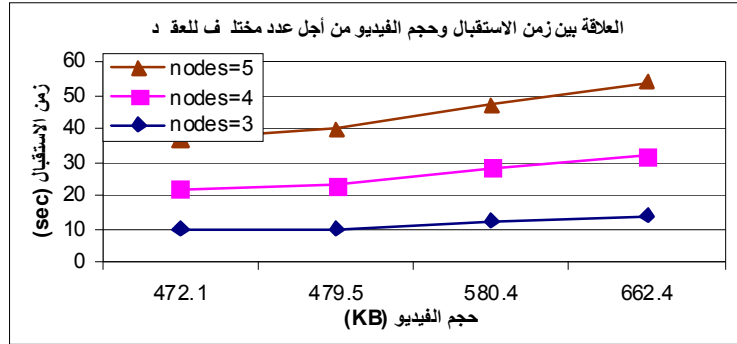
الشكل (6-22) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل عرض مجال وصلة 384Kbps.



الشكل (6-23) العلاقة بين زمن التأخير وحجم الفيديو من أجل عرض مجال وصلة 384Kbps.

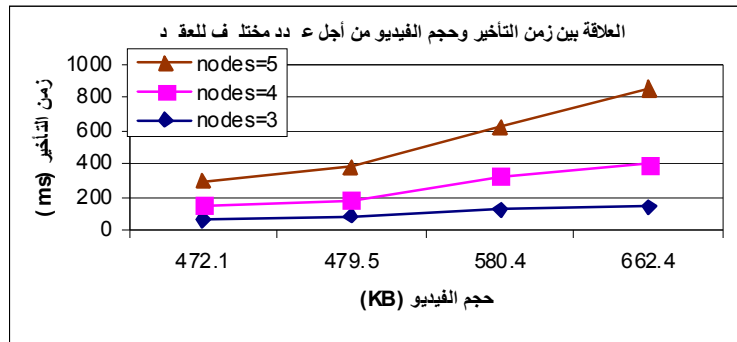
يتطلب إنقاص زمن الاستقبال وزمن التأخير للفيديو زيادة عرض مجال الوصلة الهابطة BWd ، وهذا موضح بالشكلين السابقين.

يبيّن الشكل (6-24) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل قيم مختلفة لعدد المستخدمين.



الشكل (24-6) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل عدد مختلف للعقد وعند عرض مجال وصلة يساوي 384Kbps.

ويبين الشكل (25-6) العلاقة بين زمن التأخير وحجم الفيديو من أجل قيم مختلفة لعدد المستخدمين.

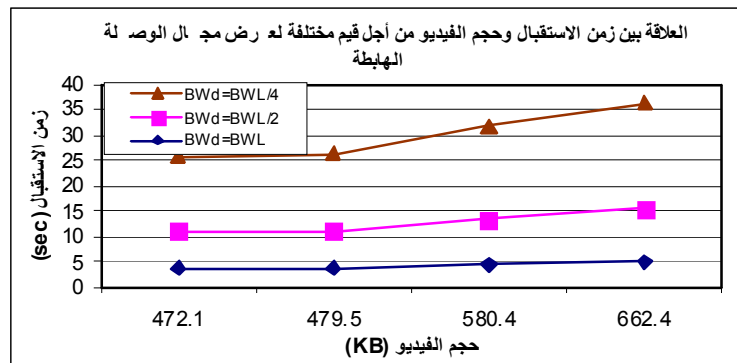


الشكل (25-6) العلاقة بين زمن التأخير وحجم الفيديو من أجل عدد مختلف للعقد وعند عرض مجال وصلة يساوي 384Kbps.

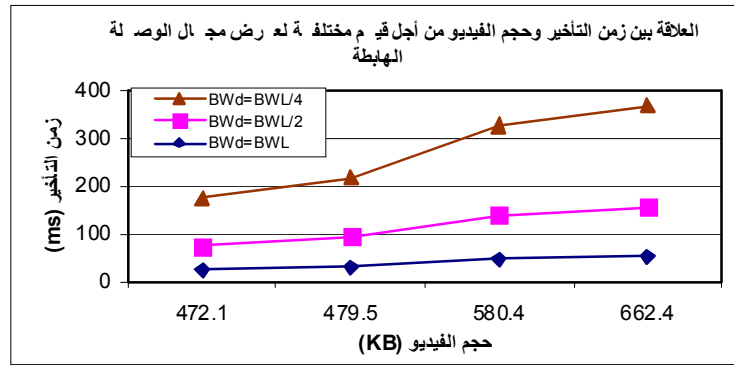
نلاحظ من الشكلين السابقين ازدياد الزمن اللازم للاستقبال وزمن التأخير مع ازدياد عدد المستخدمين بقيم مقبولة لا تؤثر بشكل كبير على أداء التطبيق.

4.5.6. دراسة متوسطات التطبيق من أجل عرض مجال وصلة 1Mbps

قمنا في هذه المرحلة، بتكرار الخطوات السابقة، وذلك من أجل عرض مجال وصلة $BWL=1Mbps$. يبين الشكلان (26-6) و (27-6) العلاقة بين زمن الاستقبال وزمن التأخير مع حجم ملف الفيديو من أجل قيم مختلفة لعرض مجال الوصلة الهابطة بالمقارنة مع عرض مجال الوصلة.



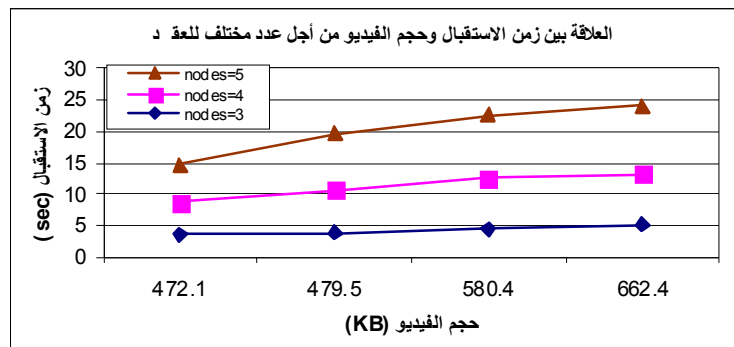
الشكل (26-6) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل عرض مجال وصلة 1Mbps.



الشكل (27-6) العلاقة بين زمن التأخير وحجم الفيديو من أجل عرض مجال وصلة 1Mbps.

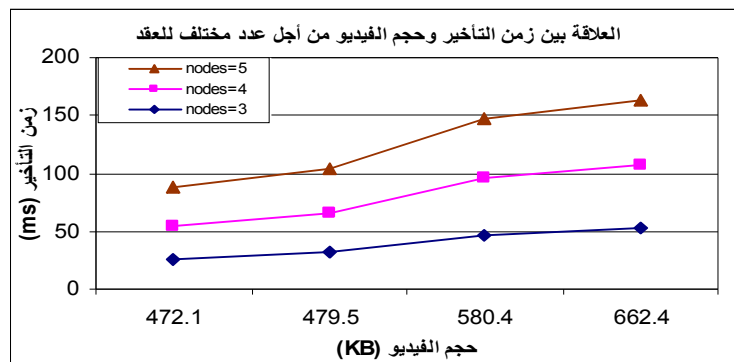
من الملاحظ ازدياد زمن الاستقبال وزمن التأخير مع إنقاص عرض مجال الوصلة الهابطة المتاحة، ولكن ذلك يؤدي إلى ازدياد عرض مجال الوصلة الصاعدة المتاحة ومن ثمَّ ازدياد عدد المستخدمين للتطبيق.

يبين الشكل (28-6) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل قيم مختلفة لعدد المستخدمين.



الشكل (28-6) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل عدد مختلف للعقد وعند عرض مجال وصلة يساوي 1Mbps.

يبين الشكل (29-6) العلاقة بين زمن التأخير وحجم الفيديو من أجل قيم مختلفة لعدد المستخدمين.

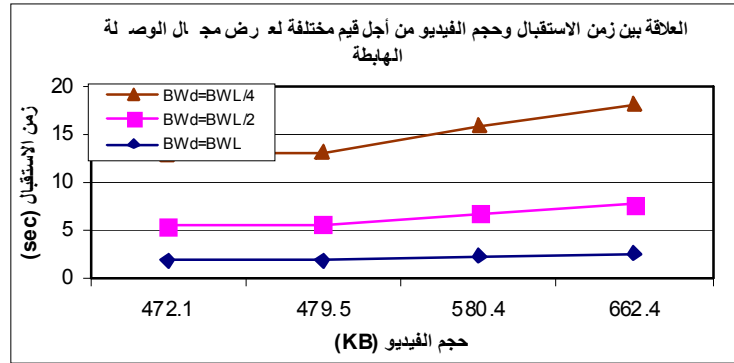


الشكل (29-6) العلاقة بين زمن التأخير وحجم الفيديو من أجل عدد مختلف للعقد وعند عرض مجال وصلة يساوي 1Mbps.

يلاحظ من الشكلين السابقين أن ازدياد عدد مستخدمي التطبيق يؤدي إلى ازدياد الزمن اللازم لاستقبال الفيديو وكذلك زمن التأخير قبل بدء عرضه بقيم مقبولة.

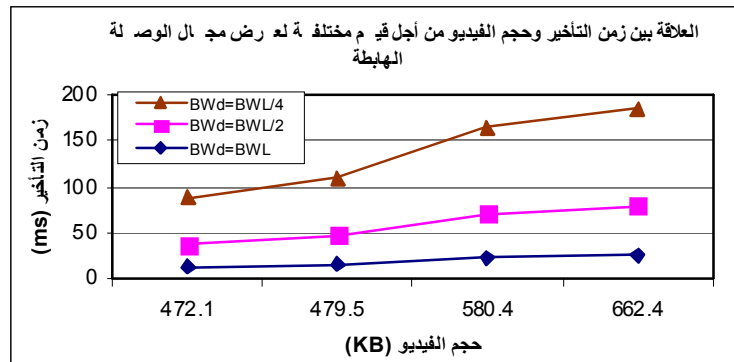
5.5.6. دراسة موسطات التطبيق من أجل عرض مجال وصلة 2Mbps

قمنا في هذه المرحلة، بتكرار الخطوات السابقة، وذلك من أجل عرض مجال وصلة $BWL=2Mbps$. يبين الشكل (6-30) العلاقة بين زمن الاستقبال وحجم الملف من أجل قيم مختلفة لعرض مجال الوصلة.



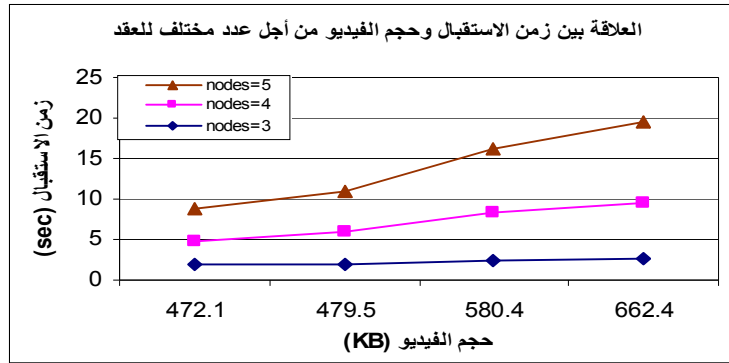
الشكل (6-30) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل عرض مجال وصلة 2Mbps.

يبين الشكل (6-31) العلاقة بين زمن التأخير وحجم الفيديو من أجل قيم مختلفة لعرض مجال الوصلة الهابطة بالمقارنة مع عرض مجال الوصلة.

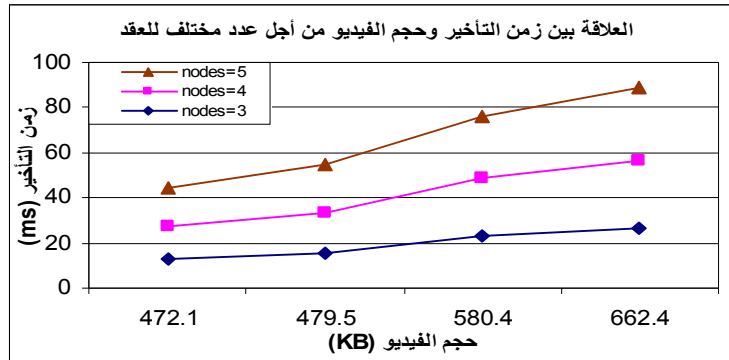


الشكل (6-31) العلاقة بين زمن التأخير وحجم الفيديو من أجل عرض مجال وصلة 2Mbps.

يتطلب إنقاص زمن الاستقبال وزمن التأخير للفيديو زيادة عرض مجال الوصلة الهابطة BWd ، وهذا موضح بالشكلين السابقين. يبين الشكلان (6-32) و (6-33) العلاقة بين زمن الاستقبال وزمن التأخير مع حجم الفيديو من أجل قيم مختلفة لعدد المستخدمين.



الشكل (32-6) العلاقة بين زمن الاستقبال وحجم الفيديو من أجل عدد مختلف للعقد وعند عرض مجال وصلة يساوي 2Mbps.



الشكل (33-6) العلاقة بين زمن التأخير وحجم الفيديو من أجل عدد مختلف للعقد وعند عرض مجال وصلة يساوي 2Mbps.

يلاحظ من الشكلين السابقين أن ازدياد عدد مستخدمي التطبيق يؤدي إلى ازدياد الزمن اللازم لاستقبال الفيديو وكذلك زمن التأخير قبل بدء عرضه بقيم مقبولة.

6.6. مقارنة التطبيق المقترح مع بعض التطبيقات الشهيرة

أجرينا في هذه المرحلة مقارنة بين التطبيق المقترح وبعض التطبيقات الشهيرة ومنها: تطبيق بت تورنت، وتطبيق غنوتلا، وتطبيق فروست واير FrostWire. ولمقارنة الأداء بين التطبيقات قمنا باستقبال ملف نصي بحجم 1.22MB باستخدام التطبيقات المذكورة. ويبين الجدول (4-6) نتائج المقارنة بين هذه التطبيقات.

الجدول (4-6) مقارنة بين التطبيق المقترح وبعض التطبيقات

التطبيق	زمن البحث	زمن بناء الشبكة (ثانية)	زمن الاستقبال (دقيقة)
بت تورنت	0	36	40
غنوتلا	غير محدد*	-	-
فروست واير	0	22	25
التطبيق المقترح	2 (ثانية)	4	15

* تطلب تطبيق غنوتلا زمن بحث كبير جداً بالمقارنة مع التطبيقات الأخرى، حيث لم يتمكن من الوصول إلى الملف المطلوب طوال فترة إجراء الاختبار.

7.6. مناقشة النتائج

بعد إجراء مجموعة من الاختبارات على أداء التطبيق، ومن أجل مستويات مختلفة توصلنا إلى النتائج الآتية:

المقارنة مع تطبيقات المخدم/زبون

- تمتاز تطبيقات الند للند بتعدد مرتفع بالمقارنة مع تطبيقات المخدم/زبون.
- يكون زمن استقبال الملفات في تطبيقات الند للند أقل منه في تطبيقات المخدم/زبون.
- في تطبيقات مشاركة الملفات والبث الفيديوي
- يتناقص زمن الاستقبال للملفات مع ازدياد وحدة عرض المجال المختارة.
- يتناقص زمن الاستقبال للملفات مع ازدياد عدد حيزات الوصلة المستخدمة (عرض مجال الوصلة).
- يزداد زمن الاستقبال للملفات مع تناقص العرض المخصص للوصلة الهابطة، وبالمقابل يزداد عدد المستخدمين للتطبيق.
- يزداد زمن الاستقبال للملفات مع ازدياد عدد المستخدمين بنسبة لا تتجاوز 13%.

في تطبيقات البث الفيديوي

- يتناقص زمن التأخير مع ازدياد عرض مجال الوصلة الهابطة المتاحة.
- يتناقص زمن التأخير مع ازدياد عدد حيزات الوصلة المستخدمة (عرض مجال الوصلة).
- يزداد زمن التأخير مع ازدياد عدد المستخدمين.

التحقق من أهداف البحث

- تصميم تطبيق برمجي لمشاركة الملفات.
- تصميم تطبيق برمجي للبث الفيديوي.
- الحصول على تصعد مرتفع للتطبيق بالمقارنة مع بنية المخدم/زبون.
- تطوير آلية للبحث عن الملفات وبناء الشبكة بزمن مهم.
- ارسال واستقبال ملفات بأنواع مختلفة بزمن لا يتجاوز 100 sec.
- التمكن من بث الفيديو بزمن تأخير لا يتجاوز 2000 ms، وذلك من أجل عرض مجال وصلة 128Kbps.

8.6. الخلاصة

عرضنا في هذا الفصل لنتائج التنفيذ العملي للتطبيق بحالتين مختلفتين وهما: الفتح بعد الإتيان بالملف، والتشغيل أثناء الإتيان به. استخدمنا للتحقق من التطبيق طريقة المحاكاة. درسنا أداء التطبيق مع تغيير عدة موسطات ومنها: حجم الملف، ووعدد حيزات الوصلة، ووحدة عرض المجال وكذلك عدد المستخدمين.

الاستنتاجات والتوصيات

1.7. ملخص البحث

تناولنا في هذا البحث تنفيذ تطبيق بث فيديو باستخدام شبكات الند للند من أجل الاستفادة من ميزات الشبكة، من حيث التصعد المرتفع، والوثوقية العالية، وسهولة التنفيذ. قمنا في بداية البحث بدراسة نظرية لشبكات الند للند، لتوضيح بنية هذه الشبكة وأهم الميزات التي دعتنا لاستخدامها، كما أجرينا مقارنة بين الأنواع المختلفة لبناء شبكة ند للند مع أمثلة عن كل نوع. تضمن البحث أيضاً استخدام شبكات الند للند من أجل مشاركة الملفات، وكذلك من أجل البث الفيديوي.

كما أجرينا مقارنة بين تقنيات بث الفيديو المعتمدة وبعض التطبيقات المستخدمة حالياً، لاستنباط التحديات الواجب أخذها بعين الحسبان عند بناء التطبيق. بعد إنجاز الدراسة النظرية المطلوبة، قمنا باقتراح بنية التطبيق وتنفيذه برمجياً باستخدام لغة سي C، ومن ثم اختبار التطبيق لدراسة أدائه في عدة حالات مختلفة والتحقق من ميزات استخدام شبكات الند للند في التطبيقات المختلفة.

2.7. مخرجات البحث research outputs

- تتضمن النقاط الأساسية التي جرت تغطيتها في هذا البحث ما يأتي:
- شرح مفصل عن شبكات الند للند المختلفة وميزات استخدام كل منها
 - شرح لبعض تطبيقات الند للند المستخدمة من أجل مشاركة الملفات
 - ملخص عن تقنيات بث الفيديو المستخدمة حالياً
 - شرح مفصل عن طريقة تنفيذ تطبيق مشاركة ملفات باستخدام شبكات الند للند بطريقة برمجية
 - شرح مفصل عن طريقة تنفيذ تطبيق بث فيديو باستخدام شبكات الند للند بطريقة برمجية
 - توصيف لبعض المكتبات البرمجية المفتوحة والمستخدم في معالجة الفيديو وعرضه
 - أداء التطبيق في حالات وبيئات مختلفة
 - تحليل نتائج اختبار التطبيق وتوثيقها

3.7. الخلاصة

تحتاج تطبيقات مشاركة الملفات والبث الفيديوي إلى عرض مجال كبير يصعب تحقيقه باستخدام التقنيات التقليدية، مثل المخدم/زبون وتقنيات البث متعدد الوجهات، وذلك بتكلفة منخفضة. أدى استخدام تطبيقات الند للند إلى تخفيض تكلفة هذه التطبيقات على الرغم من ازدياد عدد المستخدمين بشكل كبير.

نفذنا التطبيق المقترح باستخدام شبكة ند للند، وذلك للاستفادة من ميزات الشبكة بتخديم عدد كبير من المستخدمين وبتكلفة منخفضة. اخترنا البنية الهيكلية لشبكة الند للند والتي تعتمد على وجود مخدم مركزي مسؤول عن تنظيم العقد ضمن الشبكة وتسهيل عملية البحث عن الملفات. بعد إجراء عدة اختبارات للتطبيق مع تغيير مستوياته الأساسية، تحققنا من أداء التطبيق وتوصلنا إلى نتائج تبين ميزات شبكات الند للند من حيث التصعد المرتفع ووثوقية التطبيق.

4.7. المقترحات والتوصيات

جرى اختبار التطبيق باستخدام شبكة افتراضية، ومن ثمّ يمكن إجراء الاختبار ضمن شبكة حقيقية وتقييم النتائج.

كما يمكن استخدام تقنية الترميز الهرمية، والتي تعتمد على ترميز الفيديو بعدة مستويات، ومن ثمّ التحكم بعدد المستويات التي يجري إرسالها إلى المستخدم وفقاً لعرض المجال المتاح له. فعند استقبال المستخدم لكل مستويات الترميز فإنه يحصل على كامل جودة الفيديو وإلا فإن جودة الفيديو تنخفض مع انخفاض عدد المستويات المستقبلية.

الأبحاث المنشورة

- تم نشر البحث ضمن مؤتمر ICT2012 بعنوان:
"P2P application for file sharing"
- 19th International Conference on Telecommunications – Jounieh ,Lebanon
April 23 - 25, 2012 Technical Sponsor :IEEE
- تم قبول البحث بمجلة جامعة دمشق للعلوم الهندسية بتاريخ 2012/5/3 وبالعنوان:
"شبكات الند للند من أجل مشاركة الملفات"

المراجع

- [1] T. d. Laurea, "Simulation and analysis of "Interleave protocols" for distributing video streaming in peer-to-peer systems," 2007.
- [2] X. Xu, Y. Wang, S. S. Panwar, and K. W. Ross, "A peer-to-peer video-on-demand system using multiple description coding and server diversity," International conference on image processing ICIP'04, 2004.
- [3] D. Ciullo and e. al., "Network awareness of P2P live streaming applications: A measurement study," IEEE transaction on multimedia, Jan. 2010.
- [4] R. Petrocco, J. Pouwelse, and D. H. J. Epema, "Performance Analysis of the Libswft P2P Streaming Protocol," 3-5 Sept. 2012.
- [5] S. Jaganathan and J. Eranti, "High quality of service on video streaming in P2P networks using fst-mdc," in *The international journal of multimedia & its applications (IJMA)*, vol. 3, 2011.
- [6] J. Hinds, "Real-time video streaming using peer-to-peer for video distribution," University of Pretoria, 2008.
- [7] P. Baccichet, J. Noh, E. Setton, and B. Girod, "Content-aware P2P video streaming with low latency," IEEE International conference on multimedia and expo, 2-5 July 2007.
- [8] N. Efthymiopoulos, S. L. Tompros, A. Christakidis, K. Koutsopoulos, and S. Denazis, "Enabling live video streaming services realization in telecommunication networks using P2P technology," *International Journal of Communication Systems advances in multimedia communications*, vol. 24, pp. 1354-1374, october 2011.
- [9] A. Vasudeva, Sandeepan, and N. Kumar, "PASE: P2P network based academic search and file sharing application," 1st International Conference on Computational Intelligence, Communication Systems and Networks, (CICSYN'09), 2009.
- [10] S. Traverso, E. Leonardi, M. Mellia, and M. Meo, "Network awareness in P2P-TV applications," EUNICE, 2009.
- [11] B. Cohen, "Incentives Build Robustness in BitTorrent," May 2003.
- [12] S. Zehnder, "Bachelor thesis live P2P video streaming framework," University of Zurich, 2008.
- [13] F. Chen, T. Repantis, and V. Kalogeraki, "Coordinated media streaming and transcoding in peer-to-peer systems," 19th IEEE International parallel and distributed processing symposium, 2005.
- [14] D. Verma, "Content distribution networks ", Wiley, 2002.

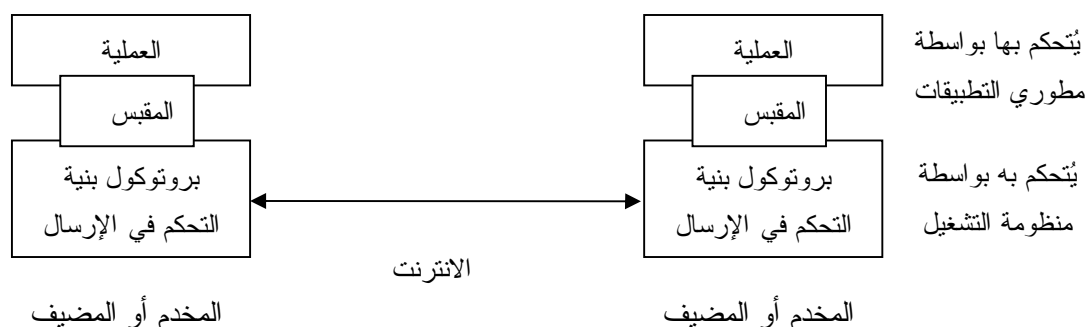
- [15] A. Magnetto, R. Gaeta, M. Grangetto, and M. Sereno, "TURINstream: A totally push, robust and efficient P2P video streaming architecture," 2010.
- [16] R. Petrocco, "Multi bit-rate video on demand for P2P networks," 2008.
- [17] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications " 1st International conference on peer-to-peer computing, (P2P'01), 2001.
- [18] K. Aberer and M. Hauswirth, "Improving data access in P2P systems," IEEE Internet computing Switzerland, 2002.
- [19] A. Marcozzi, "Introduction to the peer-to-peer paradigm," University of Bologna, 2008.
- [20] "why peer-to-peer ".
- [21] M. O. Junginger and Y. Lee, "Peer-to-Peer Middleware."
- [22] Y. Lin, "Peer-to-Peer Systems."
- [23] H. H. Tanaka, "Post-Napster: peer-to-peer file sharing systems current and future issues on secondary liability " presented at Loyola of Los Angeles futertainment law review, 2001.
- [24] "Network Applications: P2P Applications," 2008.
- [25] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," Multimedia computing and networking (MMCN), Jan. 2002.
- [26] R. Morris, M. F. Kaashoek, D. Karger, H. Balakrishnan, I. Stoica, D. Liben-Nowell, and F. Dabek, "Chord: a scalable peer-to-peer look-up protocol for internet applications," Saarland University, 2003.
- [27] J. Pang, B. Richardson, I. Stoica, M. Cuenca, B. Cohen, S. Ramesh, D. Qui, R. Shrikant, X. Li, and R. Martin, "Peer-to-Peer."
- [28] J. Liang, R. Kumar, and K. W. Ross, "Understanding KaZaA," Polytechnic University, Brooklyn, 2004.
- [29] M. Matuszewski, "Peer-to-Peer signalling," Helsinki University Of technology.
- [30] L. Garc'es-Erice, E. W. Biersack, P. A. Felber, K. W. Ross, and G. Urvoy-Keller, "Hierarchical peer-to-peer systems," *Parallel Processing Letters (PPL)*, vol. 13, pp. 643-657, 2003.
- [31] A. Kert'esz, P. Kacsuk, A. Iosup, and D. H. J. Epema, "Investigating peer-to-peer meta-brokering in grids," Technical report, TR-0170 Aug 2008.

- [32] G. Sakaryan and H. Unger, "Influence of the decentralized algorithms on topology evolution in P2P distributed networks," In design, analysis and simulation of distributed systems [DASD 2003], 2003.
- [33] M. Mushtaq and T. Ahmed, "End-to-end QoS provisioning for real time video streaming over sp-driven P2P networks using admission control," IEEE International conference on communication, ICC'09, 2009.
- [34] S. Medjiah, T. Ahmed, E. Mykoniati, and D. Griffin, "Scalable Video Streaming over P2P Networks: A Matter of Harmony?," IEEE 16th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2011.
- [35] H. K and M. D.H., "Peer-to-Peer Live Streaming and Video on Demand Design Issues and Its Challenges," International Journal of Peer to Peer Networks (IJP2P) Vol.2, No.4, October 2011.
- [36] N. Magharei and R. Rejaie, "Mesh or multiple-tree: a comparative study of live P2P streaming approaches," 2007.
- [37] M. Okuda, "Enabling large-scale peer-to-peer stored video streaming service with QoS support," University of Pittsburgh, 2006.
- [38] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A survey of application-layer multicast protocols," *Communications surveys & tutorials, IEEE*, vol. 9, pp. 58 - 74 2007.
- [39] X. Jin, W. Wong, S. G. Chan, and H. Ngan, "A survey and comparison of application-level multicast protocols," The Hong Kong University of science and technology.
- [40] D. A. Tran, K. A. Hua, and T. T. Do, "A peer-to-peer architecture for media streaming," *IEEE journal on selected areas in communications*, vol. 22, pp. 121 - 133, Jan. 2004

الملحقات

أ. برمجية المقابس *socket programming*

يتضمن هذا الملحق توضيح لمفهوم المقبس وأنواعه، إضافةً إلى التتابع الأساسية المستخدمة لبرمجة هذا المقبس. يُعدّ المقبس واجهة بين التطبيقات والشبكة، حيث يستخدم من أجل تحقيق الاتصال بين التطبيقات، كما هو مبين بالشكل (أ-1).



الشكل (أ-1) آلية الربط التي يقدمها المقبس.

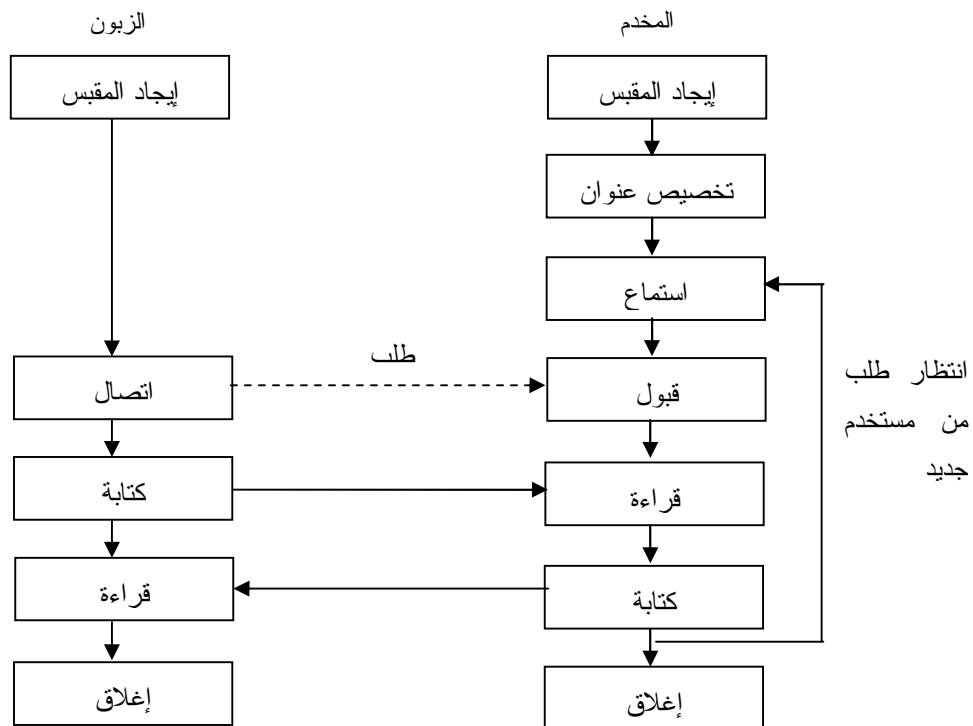
تقسم المقابس إلى نوعين أساسيين هما الدفقي Sock-stream والمخططي Sock-diagram، وبيّن الجدول (أ-1) الفروقات الرئيسية بينهما.

الجدول (أ-1) الفرق بين نوعي المقابس

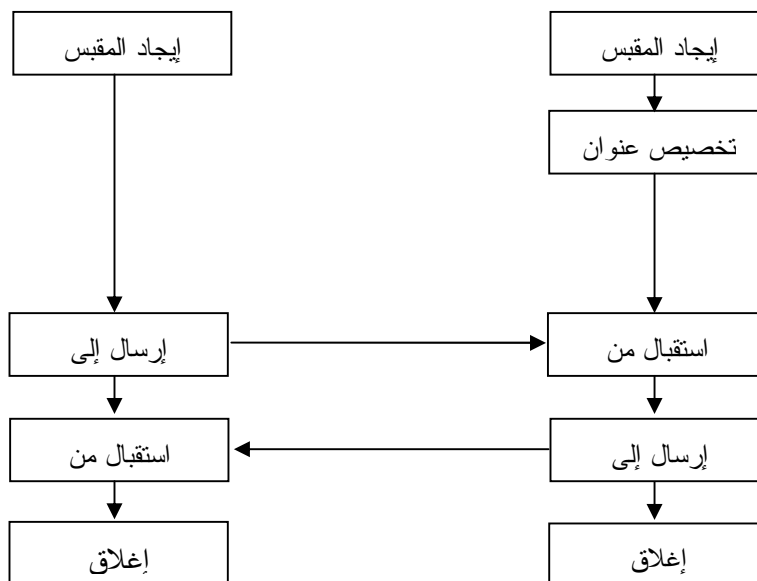
النمط المخططي Sock-diagram	النمط الدفقي Sock-stream
تعتمد على بروتوكول برقية معطيات المستخدم/الترايط الشبكي User datagram protocol/Internet protocol (UDP/IP)	تعتمد على بروتوكول بنية التحكم في الإرسال/الترايط الشبكي Transport control protocol/Internet protocol (TCP/IP)
اتصال غير موجه، حيث يجري توجيه كل رزمة على حدة	اتصال موجه connection oriented
إيصال معطيات غير موثوق unreliable delivery	إيصال معطيات موثوق reliable delivery
لا يوجد ضمان لترتيب الرزم	ضمان ترتيب الرزم in-order guaranteed
اتصال باتجاه واحد إما إرسال أو استقبال	اتصال ثنائي الاتجاه

أ.1. برمجة المقيس

يبيّن الشكل (أ-2) مراحل برمجة المقيس من أجل بروتوكول بنية التحكم في الإرسال، في حين يبيّن الشكل (أ-3) مراحل برمجة المقيس من أجل بروتوكول برقية معطيات المستخدم.



الشكل (أ-2) آلية برمجة المقيس من أجل بروتوكول بنية التحكم في الإرسال



الشكل (أ-3) آلية برمجة المقيس من أجل بروتوكول برقية معطيات المستخدم

أ.2. المكتبات الأساسية المستخدمة في برمجة المقبس

- <netinet/in.h> تستخدم من أجل تعريف البنى الأساسية
- <sys/types.h> تستخدم لتعريف أنواع المقبس
- <sys/socket.h> تتضمن التوابع الأساسية الخاصة ببرمجة المقبس
- <arpa/inet.h> تستخدم من أجل الربط بين العناوين وأسماء المضيفين host name resolution

أ.3. البنى الأساسية المستخدمة في برمجة المقبس

البنى الخاصة بالعنونة

- Struct sockaddr{ UInt8_t sa_len; Sa_family_t sa_family; Char sa_data[14]; };
حيث : Sa_family تحدد نوع العنوان و sa_data تحدد قيمة هذا العنوان.
- Struct sockaddr_in{ UInt8_t sin_len; Sa_family_t sin_family; In_port_t sin_port; Char sin_zero[8]; };
وهي بنية خاصة بتعريف المقبس من أجل IPv4
- Struct in_addr{ in_addr_t s_addr; };
• Struct hostent{ char *h_name; char **h_aliases; int h_addr_type; int h_length; char **h_addr_list; char *h_addr; };
تستخدم هذه البنية من أجل إيجاد العنوان الموافق لاسم مضيف ما.

أ.4. التوابع الأساسية المستخدمة في برمجة المقبس

توابع التحويل بين العناوين

- تحويل من صيغة محرفية إلى صيغة عنوان IPv4 وبالعكس
in_addr_t inet_addr(char * buffer)
in_addr_t inet_aton(char * buffer)
char * inet_ntoa(in_addr_t ipaddr)
- تحويل من صيغة محرفية إلى عنوان IPv6 وبالعكس
Int inet_pton(int af, const void *src, char * dst, size_t size)
- تحويل اسم المضيف إلى صيغة بنية المضيف
struct hostnet
gethostbyname()

توابع التحويل بين نمط الشبكة ونمط المضيف

Net short = htons(host short)

Net long = htonl(host long)

Host short = ntohs(net short)

Host long = ntohl(net long)

إيجاد المقبس creating socket

int socket (int family, int type, int protocol)

حيث:

Family يحدد عائلة البروتوكول ولها نوعان AF_INET من أجل الانترنت، و PF_INET

من أجل بروتوكول بنية التحكم في الإرسال/الترابط الشبكي

Type تحديد نوع الخدمة SOCK_STREAM أو SOCK_DGRAM

Protocol يحدد نوع البروتوكول (0 من أجل البروتوكول الافتراضي)

يعيد التابع رقم صحيح يمثل موصف المقبس socket descriptor أو -1 عند الخطأ.

تخصيص عنوان للمقبس assigning an address to socket

int bind (int sockfd, const struct sockaddr *my_addr, int addr_length)

يعيد التابع 0 في حال نجاح العملية و -1 عند الخطأ.

الاتصال connection

int connect (int sockfd, struct sockaddr *target, socklen_t length)

يستخدم لإنشاء الاتصال مع المخدم، يعيد 0 عند نجاح العملية و -1 عند فشلها.

الاستماع listening

int listen (int sockfd, int backlogs)

إنشاء رتل لاستقبال طلبات المستخدمين، حيث يحدد الوسط الثاني في التابع عدد الطلبات الممكن وضعها في الرتل.

القبول accept

int new_socket accept (int sockfd, struct sockaddr *client, socklen_t *client_length)

قبول طلب الاتصال وإيجاد مقبس جديد لتحقيقه، يعيد هذا التابع ضمن الوسط الثاني (client) عنوان المستخدم الذي تم قبول طلبه

إرسال المعطيات data sending

int send (int sockfd, char *buffer, size_t length, int flags)

إرسال المعطيات وإعادة عدد البتات المرسله وهو يستخدم من أجل بروتوكول بنية التحكم في الإرسال، أما من أجل بروتوكول برقية معطيات المستخدم فإنه يستعاض عنه بالتابع الآتي.

```
int sendto (int sockfd, char *buffer, size_t length, int flags, struct sockaddr *target,
            socklen_t addrlen)
```

استقبال المعطيات data receiving

```
int recv (int sockfd, char *buffer, size_t length, int flags)
```

استقبال المعطيات ووضعها ضمن الذاكرة المؤقتة وإعادة عدد البتات المستقبلية، يستخدم من أجل بروتوكول بنية التحكم في الإرسال، أما من أجل بروتوكول برقية معطيات المستخدم فيستخدم التابع الآتي.

```
int recvfrom (int sockfd, char *buffer, size_t length, int flags, struct sockaddr *target,
              socklen_t addrlen)
```

إنهاء الاتصال وإغلاق المقبس

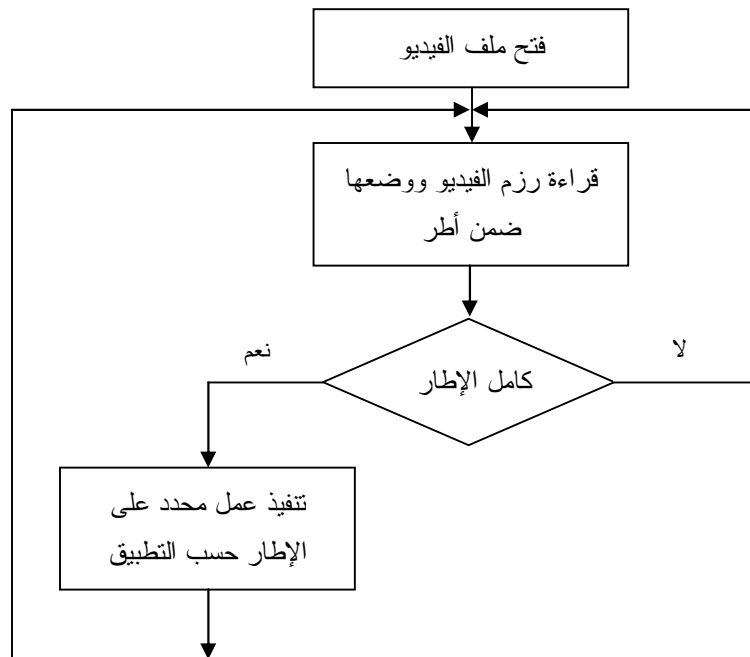
```
close (int sockfd)
```

FFMPEG ➡

يشتمل هذا الملحق على شرح موجز لمكتبة FFMPEG وأهم الأمور التي من الممكن تنفيذها، تُستخدم مكتبة FFMPEG من أجل بناء تطبيقات تسمح بتسجيل الصوت والفيديو وتحويلهما وبثهما. تتألف مجموعة FFMPEG من عدة برامج أساسية وهي:

ffmpeg	الذي يستخدم من أجل تحويل الفيديو من نمط لآخر
ffserver	يستخدم لبث الفيديو والصوت عبر الانترنت
ffplay	وهو مشغل وسائط media player بسيط مبني باستخدام مكتبة FFMPEG و SDL
libavcodec	وهي مكتبة تضم معظم مرمزات الصوت والفيديو ومفككات ترميزها المستخدمة حالياً
libavformat	مكتبة تضم مولدات لأنماط الفيديو والصوت الشائعة

يبين الشكل (ب-1) المخطط التدفقي لتصميم تطبيق فيديو أو صوت.



الشكل (ب-1) المخطط التدفقي لتصميم تطبيق فيديو أو صوت

ب.1. المكتبات الأساسية المستخدمة

<libavcodec/avcodec.h> تضم تعريف لمعظم الترميز المستخدمة

<libavformat/avformat.h> تعريف للبنى الأساسية والتوابع المستخدمة للتعامل مع ملفات الفيديو

ب.2. البنى الأساسية المستخدمة

AVFormatContext

تحتوي معلومات عن ملف الفيديو ومنها اسم الملف، المعلومات الزمنية، المؤلف، السنة، حجم الملف، ومعدل البتات، وغيرها.

AVCodec

تستخدم لتخزين معلومات الترميز المستخدم وتضم اسم الترميز، مُعرِّف الترميز، قدرات الترميز، معدلات الأطر المدعومة، أنماط البيكسلات المدعومة، وغيرها.

AVPicture

تتألف من حقلين أساسيين هما بيانات الصورة، وعدد البتات في كل سطر.

AVPacket

تتضمن معلومات الرزم بما فيها المعلومات الزمنية للزرمة، بيانات الرزم، حجم الرزمة، والمدة الزمنية للزرمة.

AVCodecContext

تستخدم لتخزين معلومات الترميز الأساسية لملف الفيديو المعالج مثل معدل البتات، أبعاد الفيديو (الارتفاع والعرض)، نمط البيكسلات، معدل العينات، عدد أقنية الصوت المستخدمة، حجم الإطار، وكذلك اسم الترميز المستخدم.

ب.3. التوابع الأساسية المستخدمة

العائد		الموسطات		التوصيف	اسم التابع
—		—		تهيئة مكتبة libavformat وتفعيل جميع الترميزات وأنماط الملفات الموجودة في المكتبات بحيث يتم استخدامها بشكل آلي عند الحاجة	av_register_all
عند النجاح	0	معلومات ملف الفيديو المعالج	ic_ptr	يقوم هذا التابع بقراءة رأس الملف وتخزين المعلومات ضمن بنية AVFormatContext	av_open_input_file
		اسم الملف المطلوب	filename		
		نمط الملف الواجب استخدامه، NULL يستخدم الافتراضي	Fmt		
فيما عدا ذلك	خطأ	الحجم الاختياري للذاكرة المؤقتة، 0 من أجل القيمة الافتراضية	buf_size		
		موسطات إضافية مطلوبة عند فتح الملف، NULL افتراضي	Ap		

العائد		الموسطات		التوصيف	اسم التابع
عند النجاح	0	معلومات الملف المعالج	Ic	إيجاد معلومات الدفق stream	av_find_stream_info
فيما عدا ذلك	خطأ				
معلومات مفكك الترميز في حال وجوده		الرقم المعرف لمفكك الترميز المطلوب	Id	إيجاد المعلومات الخاصة بمفكك الترميز وتخزينها ضمن بنية AVCodec	avcodec_find_decoder
فيما عدا ذلك	NULL				
معلومات مفكك الترميز في حال وجوده		الرقم المعرف لمفكك الترميز المطلوب	Id	إيجاد معلومات مفكك الترميز اعتماداً على اسم الترميز	avcodec_find_decoder_by_name
فيما عدا ذلك	NULL				
معلومات المرمز في حال وجوده		الرقم المعرف للمرمز المطلوب	Id	إيجاد المعلومات الخاصة بالرمز وتخزينها ضمن بنية AVCodec	avcodec_find_encoder
فيما عدا ذلك	NULL				

العائد	الموسطات		التوصيف	اسم التابع
معلومات المرمز في حال وجوده	الرقم المعرف للمرمز المطلوب	Id	إيجاد معلومات المرمز اعتماداً على اسم الترميز	avcodec_find_encoder_by_name
فيما عدا ذلك				
في حال النجاح	0	avctx	تهيئة بنية AVCodecContext بحيث تستخدم معلومات الترميز المحدد ضمن البنية AVCodec	avcodec_open
في حال الخطأ	< 0	codec		
يعيد بنية AVFrame بقيم افتراضية	—		تهيئة بنية AVFrame وملاً حقولها بالقيم الافتراضية	avcodec_alloc_frame
عند الخطأ				
يعيد بنية AVCodecContext بالقيم الافتراضية	—		تخصيص بنية AVCodecContext وملاً حقولها بالقيم الافتراضية	avcodec_alloc_context
عند الخطأ				

العائد		الموسطات		التوصيف	اسم التابع
عدد البايتات المطلوب		نمط البيكسلات	pix_fmt	حساب عدد البايتات المطلوبة لتخزين صورة بطول وعرض محددين وذلك من أجل نمط بيكسلات محدد	avpicture_get_size
		عرض الصورة	width		
		ارتفاع الصورة	height		
عند النجاح	0	ملف الفيديو المعالج	S	يقوم بقراءة الرزم ووضعتها ضمن بنية AVPacket	av_read_frame
عند الخطأ أو نهاية الملف	< 0	رزم الفيديو المقروءة	Pkt		
عدد البايتات المستخدمة		البنية المتضمنة معلومات الترميز	Avctx	فك ترميز أطر الصوت الموجودة ضمن الذاكرة المؤقتة اعتماداً على معلومات الترميز المخزنة بالبنية AVCodecContext	avcodec_decode_audio2
في حال عدم وجود إطار	0	ذاكرة الخرج	[8]samples		
		حجم ذاكرة الخرج بالبايتات	Frame_size_ptr [in,out]		
		ذاكرة الدخل	buf [in]		
		حجم ذاكرة الدخل بالبايتات	buf_size [in]		

اسم التابع	التوصيف	الموسطات	العائد
av_free	تحرير الذاكر المحجوزة	–	–
av_free_packet	تحرير الذاكر المحجوزة للرمز	الرمزة المراد تحريرها	–
av_gettime	يعطي الزمن الحالي بالمايكرو ثانية	–	الزمن الحالي بالمايكرو ثانية
avcodec_decode_video	يقوم هذا التابع بفك ترميز الفيديو المخزن ضمن ذاكرة الدخل اعتماداً على معلومات الترميز المستخدم	Avctx	بنية الترميز المستخدم
		picture	مكان تخزين إطار الفيديو بعد فك ترميزه وهو من نوع AVFrame
		buf[in]	ذاكرة الدخل
		buf_size[in]	حجم ذاكرة الدخل بالبايتات
		got_picture_ptr	يساوي الصفر في حال عدم وجود إطار يراد فك ترميزه، ولا يساوي الصفر فيما عدا ذلك
			< 0
			عند الخطأ
av_close_input_file	إغلاق ملف الفيديو بدون إغلاق بنى الترميز المستخدمة	s	بنية التابع المعالج
			–

SDL .

سنتحدث في هذا الملحق عن مكتبة الـ SDL وهي اختصار لـ (Simple Direct Layer) والتي تستخدم من أجل عرض الصور والفيديو على الشاشة.

ج.1. المكتبات الأساسية المستخدمة

والتي تضم معظم البنى والتوابع المستخدمة لعرض الفيديو
 <SDL.h>
 <SDL_thread.h>

ج.2. البنى الأساسية المستخدمة

اسم البنية	وصفها
SDL_Surface	تضم معلومات عن المكان المخصص على الشاشة لعرض الصور
SDL_Overlay	تهيئة الشاشة لتصبح مناسبة لعرض الصور
SDL_Rect	تهيئة أبعاد وأماكن توضع الإطار الذي سيتم عرض الفيديو ضمنه
SDL_AudioSpec	تستخدم من أجل تحديد مواصفات الصوت حيث تتضمن معلومات عن تردد الصوت، نمط بيانات الصوت، عدد أقنية الصوت المستخدمة، حجم الذاكرة المستخدمة لتخزين الصوت مقاسة بعدد العينات وكذلك بالبايتات
SDL_Event	تمثل البنية العامة للأحداث حيث تتضمن قائمة بتعريفات البنى الأساسية التي تتحكم بالعمل، من أهم البنى التي تتضمنها: SDL_QuitEvent quit; SDL_ResizeEvent resize; SDL_UserEvent user;

ج.3. التوابع الأساسية

اسم التابع	وصفه
SDL_Init	يقوم بتحميل المكتبات المطلوبة وتهيئة المنظومات الفرعية المحددة بموسط التابع وهي: SDL_INIT_TIMER , SDL_INIT_AUDIO , SDL_INIT_VIDEO, SDL_INIT_CDROM SDL_INIT_JOYSTICK , SDL_INIT_NOPARACHUTE, SDL_INIT_EVENTTHREAD SDL_INIT_EVERYTHING
SDL_SetVideoMode	تنصيب نمط الفيديو بالعرض والارتفاع المطلوبين وكذلك عدد البتات لكل بيكسل، يمكن التحكم بخصائص عرض الفيديو من خلال تغيير الموسط الثالث في التابع، كمثال عن ذلك: SDL_FULLSCREEN لضمان العرض على كامل الشاشة SDL_RESIZEABLE السماح بتغيير حجم الشاشة أثناء العرض
SDL_Quit	يتم تحرير كامل المنظومات الفرعية المستخدمة
SDL_LockSurface SDL_UnLockSurface	تهيئة مكان العرض من أجل الوصول المباشر للبيكسلات، يمكن بين التابعين قراءة وكتابة البيكسلات من والى مكان العرض

اسم التابع	وصفه
SDL_CreateYUVOverlay	يقوم بإنشاء مكان لعرض الفيديو
SDL_LockYUVOverlay SDL_UnlockYUVOverlay	تمكين الوصول المباشر إلى بيكسلات الشاشة
SDL_DisplayYUVOverlay	تهيئة الفيديو بحيث يتم عرضه ضمن المستطيل المحدد على الشاشة
SDL_CreateMutex	إنشاء mutex جديد
SDL_DestroyMutex	تخطيط الـ mutex
SDL_CreateCond	إنشاء متحول شرطي جديد
SDL_DestroyCond	تخطيط متحول شرطي
SDL_CreateThread	إنشاء thread جديد
SDL_OpenAudio	فتح جهاز الصوت بالموسطات المطلوبة، يعيد 0 عند نجاح العملية ويملاً المصفوفة بالمعلومات الفعلية للجهاز
SDL_PauseAudio	يستخدم من أجل تشغيل الصوت وإيقافه بشكل مؤقت يجب استدعاؤه في البداية مع متوسط يساوي 0 لبدء تشغيل الصوت
SDL_Delay	الانتظار لوقت محدد بالميلي ثانية قبل إعادة العمل
SDL_PushEvent	إضافة الحدث إلى رتل الأحداث، يعيد 0 عند نجاح العملية و -1 عندما يكون الرتل ممتلئ أو في حال الخطأ

اسم التابع	وصفه
SDL_LockMutex	وضع الـ mutex ضمن الرتل بحيث يمكن التعامل معه
SDL_UnLockMutex	تحرير الـ mutex
SDL_CondWait	الانتظار في متحول الشروط
SDL_WaitEvent	الانتظار حتى الحدث المتاح التالي يعيد 1 أو 0 عند الخطأ
SDL_CloseAudio	إيقاف عملية تشغيل الصوت وإغلاق جهاز الصوت
SDL_AddTimer	إضافة مؤقت جديد إلى مجموعة المؤقتات التي تعمل حالياً ويعيد معرف المؤقت ID أو NULL عند الخطأ

د. البرمجة بلغة سي

سنحدث في هذا الملحق عن لغة C والتي تُعدّ لغة برمجة عالية المستوى. جرى تطوير لغة سي C في العام 1972 بواسطة دينس ريتشي Dennis Ritchie في مختبرات بيل Bell. تمتاز لغة سي C بالعديد من الميزات وهي:

- لغة عامة: أي تصلح لعمل برامج قواعد البيانات، الرسومات، الحسابات، ومنظومات التشغيل.

- لغة بنوية: يكون البرنامج المكتوب بلغة سي C عبارة عن تابع رئيسي يستدعي مجموعة من التوابع الأخرى.

- تتعامل على مستوى البت: حيث يمكنها القيام بعمليات على مستوى البت.

- لغة متنقلة portable: إمكانية التعامل مع أجهزة مختلفة مثل IBM و Apple وغيرها.

د.1. البنية الأساسية لبرنامج سي C

```
#include <stdio.h>    //تضمين المكتبات والملفات الرأسية
int main()            //التابع الرئيسي
{
    //جسم البرنامج
    return 0;          //عائد التابع
}
```

د.2. أنواع البيانات

يبين الجدول (د-1) أنواع البيانات المستخدمة بلغة سي C بالإضافة لعدد البايتات التي تشغلها

الجدول (د-1) أنواع البيانات المستخدمة بلغة سي C مع عدد البايتات التي يشغلها كل نوع

النوع	الحجم (بايت)	مجال القيم
char, signed char	1	128- إلى 127
char, unsigned char	1	0 إلى 255
int, signed int	2	32,768- إلى 32,767
int, unsigned int	2	0 إلى 65535
signed short int	2	32,768- إلى 32,767
unsigned short int	2	0 إلى 65535
signed long int	4	2,147,483,648- إلى 2,147,483,647
unsigned long int	4	0 إلى 4,294,967,295

د.3. المؤثرات

تعرف المؤثرات operators على أنها الرموز التي تربط بين المتغيرات والثوابت لتحقيق معادلة ما، وتقسم إلى عدة أنواع.

المؤثرات الحسابية arithmetic operators

+	الجمع
-	الطرح
*	الضرب
/	القسمة
%	باقي القسمة

مؤثرات المقارنة relational operators

>	أكبر من	greater than
<	أصغر من	less than
==	يساوي	equal to
!=	لا يساوي	not equal to
<=	أصغر من أو يساوي	less than or equal to
>=	أكبر من أو يساوي	greater than or equal to

المؤثرات المنطقية logical operators

&&	And
	Or
!	Not

مؤثرات الزيادة والنقصان increment and decrement operators

++	زيادة بمقدار واحد
--	نقصان بمقدار واحد

د.4. توابع الإدخال والإخراج

تابع الطباعة على الشاشة printf()

تُستخدم لطباعة البيانات بجمع أنواعها على الشاشة وهي متضمنة بمكتبة <stdio.h>، ويمثل الجدول (د-2) الرموز المستخدمة مع تابع الطباعة.

الجدول (د-2) الرموز المستخدمة مع تابع الطباعة

الرمز	عمله	مثال
\n	الانتقال لسطر جديد	printf("\n")
\t	نقل المؤشر لبعد 8 مسافات (tab)	printf("\t")
\b	ارجاع المؤشر للخلف مسافة واحدة (backspace)	printf("\b")
\xdd	طباعة الحرف المقابل للكوند المكتوب بالنظام السداسي عشري hexadecimal	printf("\xdd") => a
\ddd	طباعة الحرف المقابل للكوند المكتوب بالنظام الثماني octal	printf("\101") => a

أما من أجل طباعة قيم المتغيرات تستخدم رموز معينة لتحديد نوع البيانات المراد طباعتها، كما هو مبين بالجدول (د-3).

الجدول (د-3) الرموز المستخدمة في تابع الطباعة من أجل طباعة المتغيرات على الشاشة

الرمز	عمله	نوع المتغير	مثال
%d	توصيف لمتغير رقمي صحيح	int	printf("%d",10)
%f	توصيف لمتغير حقيقي	float	printf("%f",507)
%c	توصيف لمتغير محرفي	char	printf("%c","a")
%s	توصيف لعبارة محرفية	string	printf("%s","Hello")
%u	توصيف لمتغير رقمي صحيح بدون إشارة	unsigned int	printf("%u",1)
%x	توصيف لمتغير بالنظام السداسي عشري	hexadecimal	printf("%x",af)
%o	توصيف لمتغير بالنظام الثماني	octal	printf("%o",67)

تابع الإدخال scanf()

تستخدم لإدخال جميع أنواع البيانات وهي تأخذ نفس المعاملات المستخدمة في تابع الإخراج.

توابع الإدخال والإخراج الخاصة

getchar ، putchar تستخدم لإدخال وإخراج حرف واحد فقط
 gets ، puts تستخدم لطباعة وإدخال سلسلة حرفية. وهي تختلف عن printf و
 scanf بأنها لا تستطيع التعامل مع متغيرات رقمية

د.5. أوامر التحكم

وهي تقسم إلى الحلقات loops والتفرع branches

الحلقات يبين الجدول (د-4) أنواع الحلقات وعمل كل منها.

الجدول (د-4) أنواع الحلقات وعمل كل منها.

الحلقة	وصفها	صيغتها
For	تستخدم لتكرار تنفيذ تعليمة أو أكثر عدد محدد من المرات	(مقدار الزيادة ; شرط إنهاء التكرار ; القيمة الابتدائية) for { مجموعة تعليمات }
While	تكرار مجموعة من التعليمات حتى اختلال الشرط ضمن الحلقة	{ مجموعة تعليمات } (شرط) while
do while	تكرار مجموعة من التعليمات بناءً على شرط معين، حيث يجري تنفيذ التعليمات التي تلي do ثم اختبار الشرط، في حال تحققه يجري تكرار التعليمات وإلا يوقف تنفيذ الحلقة	(شرط) while { مجموعة تعليمات } do

التفرع تغيير مسار البرنامج، يكون التفرع إما مشروطاً أو غير مشروط

التفرع المشروط:

If: تستخدم لتنفيذ تعليمة واحدة أو أكثر وفقاً لشرط معين، حيث يجري تنفيذ التعليمات فقط في حال تحقق الشرط.

{ مجموعة تعليمات } (شرط) if

If..else: تستخدم لتنفيذ أحد اختياريين، ففي حال تحقق شرط معين يجري تنفيذ مجموعة من التعليمات وإلا يجري تنفيذ مجموعة أخرى من التعليمات.

```

{ مجموعة تعليمات 1 } (شرط) if
else { مجموعة تعليمات 2 }

```

switch...case: تستخدم من أجل مقارنة متغير ما مع مجموعة قيم ثابتة وتنفيذ التعليمات التي توافق الاختيار الصحيح

```

switch(variable){
    case value1: commands 1; break;
    case value2: commands 2; break;
    default: commands 3; break; }

```

التفريع غير المشروط goto

الانتقال إلى مكان محدد داخل البرنامج بدون شروط

```

#include <stdio.h>
int main() {
    ss: printf("%s\n","Hello");
    printf("%s\n","World");
    goto ss;
    return 0; }

```

د.6. المصفوفات

وهي مجموعة من البيانات المتشابهة بالنوع والتي تملك اسم واحد. عند الإعلان عن مصفوفة فإنه يجب تحديد ثلاثة نقاط أساسية وهي:

- اسم المصفوفة
- نوع البيانات ضمنها
- عدد عناصر المصفوفة

مثال: للإعلان عن مصفوفة من 10 أعداد صحيحة باسم a نكتب:

```
int a[10];
```

كما يمكن اعطاء قيم ابتدائية لعناصر المصفوفة كما يأتي:

```
int a[4] = {10,12,23,40};
```

عند الحاجة إلى استدعاء عنصر ما فإنه يجري ذكر ترتيب العنصر ضمن المصفوفة كما يأتي:

`a[2] => 12`

د.7. السلاسل المحرفية

سلسلة من الأحرف أو الرموز أو الأرقام، يمكن التعامل مع السلاسل المحرفية باستخدام المصفوفات، حيث يجري تعريف مصفوفة محرفية بحجم هو الحد الأقصى لعدد المحارف التي يمكن إدخالها.

`char text[14] = "Hello World!";`

دوال التعامل مع السلاسل المحرفية

يبين الجدول (د-5) أهم الدوال المستخدمة مع السلاسل المحرفية ومثال عن كل منها.

الجدول (د-5) أهم التوابع المستخدمة مع السلاسل المحرفية.

التابع	العمل	مثال
gets	تستخدم لإدخال النصوص من لوحة المفاتيح	<code>char name[255]; gets(name);</code>
strcpy	نسخ ولصق الأحرف من سلسلة لأخرى	<code>strcpy(string1,string2)</code>
strncpy	نسخ ولصق عدد محدد من الأحرف من سلسلة لأخرى	<code>strncpy(string1,string2,4)</code>
Strcat	إضافة النص المخزن في السلسلة الثانية إلى نهاية السلسلة الأولى	<code>strcat(string1,string2)</code>
strncat	إضافة عدد محدد من أحرف النص المخزن في السلسلة الثانية إلى نهاية السلسلة الأولى	<code>strncat(string1,string2,4)</code>
puts	تستخدم لطباعة عبارة محرفية على الشاشة	<code>char name[255]; puts(name);</code>

د.8. المؤشرات

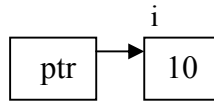
تستخدم المؤشرات pointers للإشارة إلى عنوان في الذاكرة، ويعلن عنها كما يأتي:

`pointer_type *pointer_name;`

`pointer_type`: يحدد نوع المؤشر وهو نوع المتغير الذي يشير إليه `double, float, int, char`

*: يميز المؤشر عن المتغير

pointer_name: اسم المؤشر



مثال:

```
int *ptr, i=10;
ptr = &i;
```

د.9. التتابع

يُعرف التابع function على أنه مجموعة من الأوامر والبيانات لها اسم واحد، حيث يمكن استدعائه ضمن البرنامج الأساسي عدة مرات في أماكن مختلفة.

```
function_type function_name (parameters){
Commands;
Return ;}
```

حيث :

function_type نوع القيمة التي يعيدها التابع باستخدام تعليمة return وهي نفس أنواع المتغيرات

function_name اسم التابع والذي يستخدم لاستدعائه ضمن البرنامج

Parameters الوسائط التي تمرر إلى التابع من البرنامج الأساسي وذلك لتنفيذ أوامر

محددة عليها. كما يمكن عدم تمرير أي وسائط حيث تستخدم void

Commands مجموعة الأوامر التي تنفذ ضمن التابع

د.10. الإجراءات

تشبه الإجراءات procedures الدوال (التتابع) ولكنها لا تعيد أي قيمة، حيث تكون من نوع

.void

د.11. التعامل مع الملفات

إنشاء الملف أو فتحه

في البداية يجري إنشاء مؤشر للبنية FILE والذي يستخدم في الدوال الخاصة بالملفات.

```
FILE *file_out;
```

ثم يجري إنشاء أو فتح الملف كما في الشكل الآتي

```
file_out = fopen(char *file_name, mode);
```

يمثل الوسيط الأول اسم الملف، بينما يمثل الوسيط الثاني نمط استخدام الملف كما هو مبين في الجدول (د-6).

الجدول (د-6) أنماط التعامل مع الملفات.

النمط	الوصف
"r"	فتح الملف من أجل القراءة، بشرط أن يكون الملف موجود
"w"	إنشاء ملف فارغ من أجل الكتابة، في حال كان الملف موجود فإنه يتم حذفه
"a"	إضافة المعطيات إلى نهاية الملف في حال وجوده، أما في حال عدم وجود الملف فإنه يجري إنشاؤه
"r+"	فتح الملف للقراءة والكتابة، في حال وجود الملف فإنه تجري كتابة المعطيات الجديدة في بداية الملف فوق المعطيات القديمة
"w+"	فتح الملف للقراءة والكتابة، في حال وجود الملف فإنه يجري حذف المعطيات القديمة بالكامل وكتابة المعطيات الجديدة مكانها
"a+"	فتح الملف للقراءة والإلحاق appending، في حال وجود الملف فإنه يجري إضافة المعطيات في نهاية الملف

إغلاق الملف

```
fclose(FILE *file_out);
```

القراءة من الملفات

```
fgets(char * string, int size, FILE *file_out);
```

حيث تجري قراءة سلسلة من المحارف من الملف بحجم محدد ضمن التابع وتخزينها ضمن المتغير المطلوب.

```
fscanf(FILE *file_out, "%d", int &var);
```

تستخدم أيضاً للقراءة من الملف وتخزين المعطيات بالنمط المحدد في المتغير المطلوب

الكتابة في الملفات

```
fputs(char *string, FILE *file_out);
```

تستخدم لطباعة نص محدد ضمن الملف

```
fprintf(FILE *file_out, "%s%d", "Hello", 255);
```

تستخدم من أجل طباعة نص منسق ضمن الملف

التنقل ضمن الملفات

```
fseek(FILE *file_out, int byte, mode);
```

يستخدم من أجل نقل المؤشر عدد من البايتات بدءاً من مكان المؤشر وذلك وفقاً للنمط المعطى، كما يأتي:

SEEK_SET: وضع المؤشر في بداية الملف

SEEK_CUR: الموقع الحالي للمؤشر

SEEK_END: وضع المؤشر في نهاية الملف

د. 12. البنى

مجموعة من المتغيرات تعرف باسم واحد، وهي تختلف عن المصفوفات بأنها تضم متغيرات بأنواع مختلفة.

```
struct struct_name {variables & constants};
```

يمكن التعامل مع حقول البنية كما هو موضح في المثال الآتي

```
struct student {
char *student_name;
int age;
float average;
```

```
};  
int main(){  
    struct student student1;  
    strcpy(student1.name,"Ahmad");  
    student1.age = 22;  
    student1.average = 78.5;  
    return 0;}
```

كما يمكن التعامل مع البنى كأبي نوع من المتغيرات، حيث يمكن تعريف مصفوفة من البنى وكذلك مؤشرات عليها:

```
struct student student1[20];  
struct student *sptr;
```

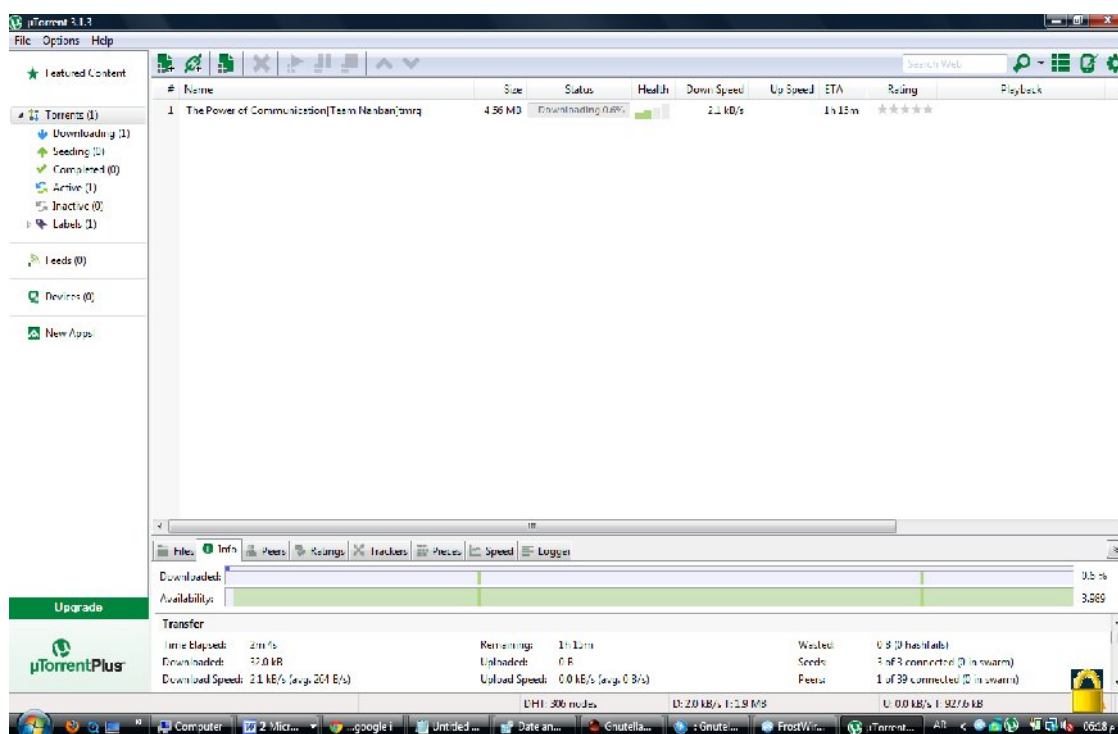
هـ. بعض تطبيقات الذكاء للذكاء الصغيرة

1.5. تطبيق ميو تورنت

يعد تطبيق ميو تورنت µtorrent مماثلاً لتطبيق بت تورنت، تم تصميمه من قبل لودفيغ ستريجوس Ludvig Strigeus، ومن ثمَّ طور بواسطة فريق من المصممين بشركة بت تورنت BitTorrent Inc. بعض الميزات الأساسية

- تحديد عرض المجال
- دعم جدول التشفير الموزع DHT
- شريط تنزيل Download bar
- جدولة التنزيل download scheduler
- متعقبات مضمنة Embedded tracker
- دعم متعقبات HTTPS
- دعم لائحة حجب العناوين
- شريط بحث Search bar

يبين الشكل (1-5) واجهة تطبيق ميو تورنت µtorrent



الشكل (1-5) واجهة تطبيق ميو تورنت µtorrent

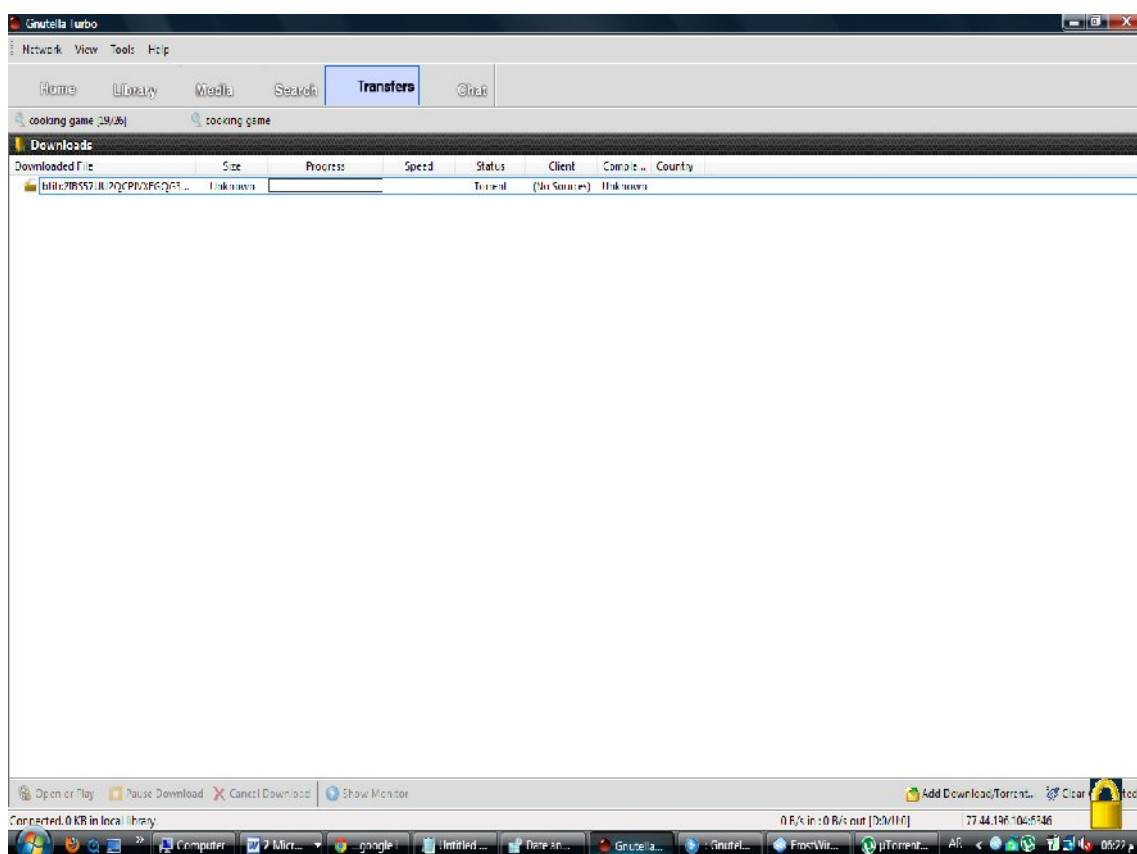
ه.2. التطبيق غنوتلا تربو

يعدّ التطبيق غنوتلا تربو Gnutella Turbo تطبيق ند لند من أجل مشاركة الملفات مصمم من أجل عملية التنزيل الموثوقة. يستخدم من أجل جميع أنواع الملفات الصوتية mp3، الفيديو، الصور، والتطبيقات، كما يستخدم من قبل أي مستخدم للانترنت.

بعض الميزات الأساسية

- يجمع بين سهولة الاستخدام والوثوقية
- إمكانية متابعة التنزيل بعد الانقطاع
- عرض مجال محدد للتنزيل والتحميل
- استخدام المرشحات
- إمكانية البحث المتعدد

يبين الشكل (ه-2) واجهة التطبيق غنوتلا تربو Gnutella Turbo



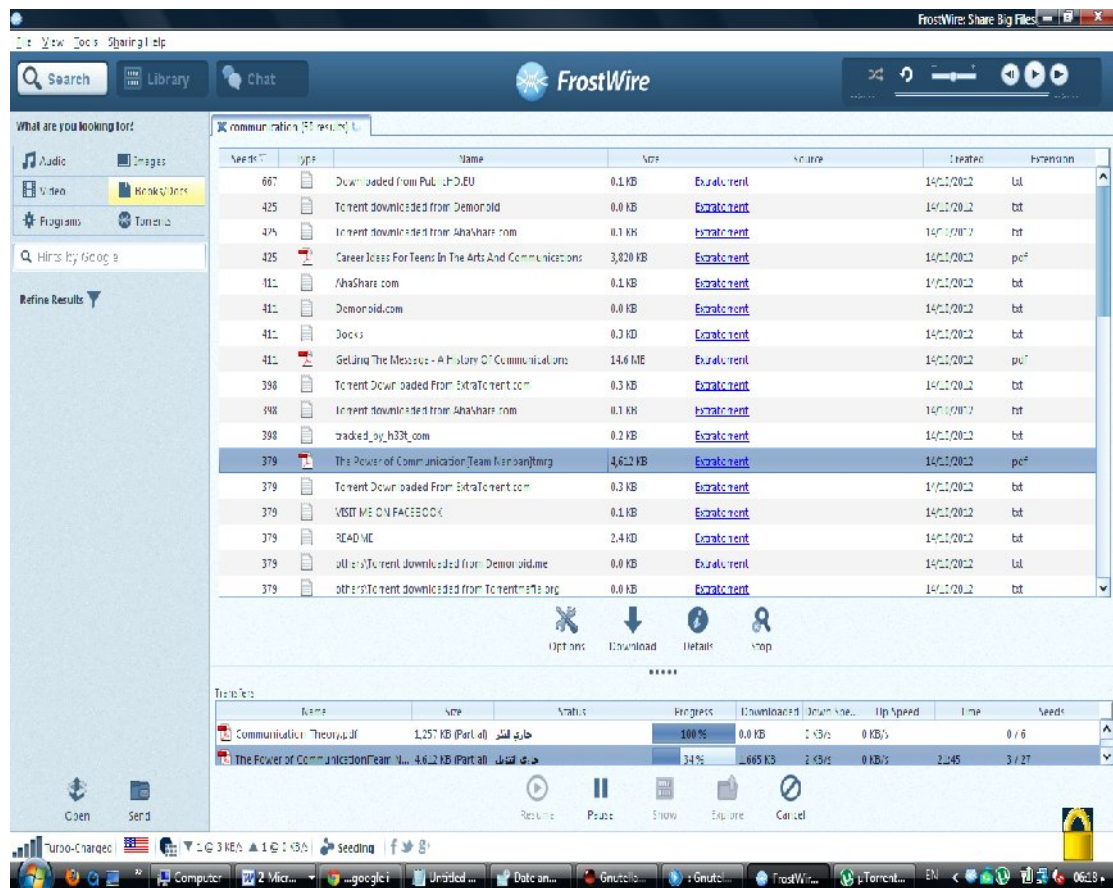
الشكل(ه-2) واجهة التطبيق غنوتلا تربو Gnutella Turbo

3.5. فروست واير

يمثل تطبيق فروست واير FrostWire في مبدأ عمله تطبيق بت تورنت، وهو يستخدم لمشاركة الملفات كبيرة الحجم. يتميز بإمكانية تبادل الملفات ضمن شبكة لاسلكية محلية بين مجموعة حواسيب أو أجهزة نقالة أو لوائح مكتبية tablet.

بعض الميزات الأساسية

- مشغل فيديو خاص
 - إمكانية سماع الصوت عند البحث
 - إمكانية استعراض الفيديو خلال البحث
 - إمكانية مشاركة اللاسلكية بين الحواسيب واللوائح المكتبية Android
- يبين الشكل (3-5) واجهة التطبيق فروست واير FrostWire



الشكل (3-5) واجهة التطبيق فروست واير FrostWire

Abstract

The advantage of the peer-to-peer (P2P) paradigm relies on two main concepts: cooperation among users and resource sharing. There are many applications based on peer-to-peer paradigm, like file sharing and video streaming applications. We can classify the file sharing application into centralized systems, (having a central server), and decentralized systems. Another classification would be structured and unstructured systems, based on the way of managing the indexing information.

In video streaming application, the video starts automatically after receiving small part of it. In contrast, we have to wait until the file finished downloading before using it.

In this research we have implemented a centralized peer-to-peer application for file sharing and for video streaming. We implemented the system using C language and evaluated the performance.

Acknowledgments

Foremost, I would like to express my sincere gratitude to my supervisor, Associate Professor Mohamed Khaled Chahine, from The University of Damascus. He provided me with direction, technical support, encouragement, sound advice, and a lot of good ideas.

His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my master study.

My sincere thanks also go to Professor Gianluca Mazzini, from the University of Ferrara, for the continuous support of my master study and research, for his patience, motivation, and immense knowledge.

It was through his, persistence, understanding and kindness that I completed my master degree.

Dedication

*I would like to dedicate this master thesis to **my family**. There is no doubt in my mind that without their continued encourage and support I could not have completed this work.*

*This thesis is, also, dedicated to **my friends and colleagues**, who helped and encouraged me throughout my study.*

Finally, I dedicate this thesis to everyone who has lent a hand in this project.