Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A cross-layer middleware for context-aware cooperative application on mobile ad hoc peer-to-peer network

Jun-Li Kuo*, Chen-Hua Shih, Cheng-Yuan Ho, Yaw-Chung Chen

Information and Communications Research Laboratories (ICL), Industrial Technology Research Institute (ITRI), Hsinchu, Taiwan

ARTICLE INFO

Article history:
Received 27 September 2012
Received in revised form
25 September 2013
Accepted 2 October 2013
Available online 11 October 2013

Keywords:
Mobile wireless P2P
Cross-layer middleware
IPv6
Anycast
Routing header
Mobile ad hoc network

ABSTRACT

Mobile ad hoc peer-to-peer (P2P) applications become popular for providing the file sharing, voice communicating, and video streaming services due to entertainments and disaster recovery. However, both the topology of wireless network and the overlay of P2P network are dynamic, so the middleware is proposed to integrate such architectures of service-oriented applications. Therefore, we propose context-aware cooperative application (CACA) to overcome the frequent churn and high mobility problems. CACA proposes a cross-layer middleware to integrate DHT-based lookup, anycast query, and P2P delivery via the IPv6 routing header. Through anycast query, the response delay can be shortened and the query duplication can be minimized. Via IPv6 routing header, the delivery efficiency can be improved. Through the cross-layer design, the finger table in overlay layer is combined with the routing table in network layer to heighten proximity. The simulation results demonstrate that CACA has the outstanding performances of short download delay, high playback continuity, and low signaling overhead in mobile ad hoc network.

1. Introduction

With the development of wireless broadband and mobile technique, the arbitrary collaboration service or content sharing application can be expanded successfully. Wireless local area network (WLAN) provides both Internet accessibility and ad hoc availability for mobile devices. For example, two famous WLAN protocols, WiFi and WiMax, can support network capability without infrastructure in the ad hoc mode, which is generally introduced mobile ad hoc network (MANET) (Sesay et al., 2004).

In MANET, two end nodes can communicate with each other through other intermediate nodes over wireless infrastructure-less network without the centralized mechanism. The nodes organize a cooperative network by themselves to establish communications in MANET with low setup cost and short startup delay. Such mobile nodes can share resources and files via the peer-to-peer (P2P) scheme due to the innate similarities.

Like a peer of P2P application, a mobile node simultaneously plays the roles of server, router, and client. When a mobile node is the server, it may be an agent, an anchor, or a repository of WLAN, and becomes the source to provide data. When a mobile node is the router, it may be an intermediate node on the communication, and becomes the relay to forward data. When a mobile node is the client, it may be an end user or a receiver, and becomes the terminal to query and download data.

Like MANET, P2P network builds the service-oriented overlay to share data. Peers always collaborate with each other without any centralized server. Therefore, how to keep the scalability and how to deliver data efficiently are both important to both P2P and mobile networks. In order to achieve high scalability and efficiency, more and more P2P applications have been implemented in wireless network in recent years, such resolutions can be called mobile P2P, wireless P2P, or P2P over MANET. For example, COME-P2P (Kuo et al., 2012) presents a cross-layer design to deliver P2P live multimedia streaming in MANET.

In this paper, we propose a modification of COME-P2P to extend not only live streaming but also file sharing services. The modification integrates the cross-layer design with IPv6 route inherited from COME-P2P into a middleware. The proposed middleware, context-aware cooperative application (CACA), combines the context-aware search, the cooperative data management, the anycast query, and the mobile P2P delivery. CACA is independent from upper layer for P2P applications and lower layer for MAC protocols, so it is extendable for all kinds of P2P sharing services with scalability and mobility. Based on IPv6, CACA proposes a comprehensive mobile P2P scheme from query to search, maintenance, and delivery.

The paper is organized as follows: Section 2 presents the related works. Section 3 introduces CACA algorithm and middleware scheme including of DHT-based overlay, anycast query, and cross-layer design. Section 4 presents the performance of compared

* Corresponding author. Tel.: +886 35731851.
E-mail address: estar.cs95g@nctu.edu.tw (J.-L. Kuo).

0164-1212/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.10.007

algorithms via the mathematical analysis. Section 5 discusses the simulation results of file sharing and live streaming. Section 6 concludes the paper.

2. Related works

Because CACA involves P2P overlay, mobile network, and IPv6 protocol, we discuss the mobile P2P scheme, middleware design, IPv6 routing header, and IPv6 anycast one by one and survey the recent works.

2.1. Mobile P2P scheme

Mobile P2P network has become a popular and interesting development of file sharing and video streaming so far. A general mobile P2P scheme should consider both P2P sharing protocol and mobile routing protocol. The *finger table* is used to search peer and data in overlay layer, and the *routing table* is used to route packets in network layer. Every peer maintains the finger table to locate its neighbors and forward data, so P2P overlay is self-organized via the finger table. The item has a one-to-one mapping of the destination and the next hop in the finger table, and *distributed hash table* (DHT) is used to index peers and files in the overlay, such that it hastens the query process and heightens the content availability.

Based on the above background, mobile P2P scheme should consider the following issues:

- *Scalability*: A mobile P2P application usually is not developed for particular users. On *logical overlay*, the size of P2P service varies constantly due to peer churn problem. On *physical topology*, the size of wireless area also varies due to the self-organized infrastructure-less environment. Therefore, the mobile P2P scheme must be robust and scalable for the dynamic user space.
- *Recoverability*: The peers in mobile P2P network leave and join arbitrarily. The word *churn* describes the behavior, or means that peers arrive and depart at a high rate. The peer churn is often unexpected a priori, and it destroys delivery overlay and breaks routing path. Therefore, the mobile P2P scheme must recover the overlay to restore the service.
- *Mobility*: A major difference between wired P2P and mobile P2P is the mobility. Therefore, the mobile P2P scheme must use a mobility detection function to monitor peers' movement and check peers' departure. The mobility detection includes the *active* notification and the *reactive* notification.
- *Adaptability*: P2P overlay is self-organized independently from routing topology, which is also self-organized without any central administration or control. The wired P2P forwarding protocol leads to the far routing problem due to a lack of overlay proximity, i.e. an inefficient organization leads to an inefficient P2P traffic. Therefore, the mobile P2P scheme must be adaptable for dynamic environment.
- *Extensibility*: How to integrate with P2P overlay and routing topology to deal with a general purpose is important in the mobile P2P scheme. The extensibility describes the comprehensive integration of P2P application on mobile device, but the mobile P2P scheme often inherits from wired P2P scheme. Therefore, the mobile P2P scheme must improve the extensibility to heighten the popularity and flexibility.
- *Interoperability*: The mobile P2P applications should be service-oriented, but they are usually developed for a specific service in the heterogeneous wireless network. The mobile P2P application usually has an innate hardware limitation and binds the wireless technology, such that the heterogeneity limits the interoperability. Therefore, the mobile P2P scheme must provide a platform to heighten the interoperability.

For example, a wired DHT-based P2P protocol is unsuitable for mobile network due to a lack of integrated consideration. CAN (Ratnasamy et al., 2001) and Chord (Stoica et al., 2001) have been used successfully in wired P2P network, but the design is unsuitable for mobile network due to a lack of recoverability and mobility. Therefore, M-CAN (Mobile-CAN) (Peng et al., 2004) and M-Chord (Mobile-Chord) (Li et al., 2006) modify the DHT to be suitable for mobile network.

2.2. Middleware design

In order to achieve the scalability, recoverability, mobility, adaptability, extensibility, and interoperability, the middleware between application layer and MAC layer is proposed to provide a mobile P2P platform. The middleware is an integrated dependency of overlay tier and routing tier, and it is independent from P2P service and wireless technology. The different kinds of P2P applications and wireless accessibilities are workable and compatible under the middleware.

For example, MAPCP (MANET Anonymous P2P Communication Protocol) (Chou et al., 2007) is a middleware between the P2P applications and ad hoc routing protocols. MAPCP consists of two phases and two tables: a *destination table* records the results of query in the *query phase*, and a *path table* records the information of multiple paths in the *data transmission phase*. P2P-HWMP (P2P over Hybrid Wireless Mesh Protocol) (Haw et al., 2009) exploits the route discovery functionality and information provided by the underlying routing layer to improve the P2P search performance, and it collates the routing information to get the overlay proximity.

COME-P2P (Cross-layer Overlay for Multimedia Environment on wireless ad hoc P2P) (Kuo et al., 2012) motivates the live streaming for high data rate and time sensitivity on mobile network. It integrates both *enhanced DHT* (EDHT) lookup and IPv6 route to improve the delivery efficiency. The hop-by-hop routing path for overlay proximity is derived via the *path information*, which is altered to accommodate the mobility and the changing topology according to the latency. CACA can be seen as an improvement of COME-P2P, so we introduce IPv6 routing header in following section.

2.3. IPv6 routing header

The extended *routing header* in IPv6 helps forward packets hop by hop in pre-computed order, such that the attribute "next hop" of IPv6 is used to forward packets to the destination. Every mobile node can check the *destination* and the *segment left* of IPv6 packet to determine the next hop. As Fig. 1 illustrated, in COME-P2P, when a logical P2P path $A \rightarrow D \rightarrow E$ is established on P2P overlay via EDHT, the path information is updated in EDHT and used to decide the next hop in each peer. The path information is filled in the attribute "routing header" of IPv6 packet, and then every mobile node checks the destination to forward it to the next node.

2.4. Anycast

Besides routing header speeds up and stabilizes data delivery, anycast is another characteristic of IPv6 to reduce the querying overhead. Anycast is an addressing or routing method based on IP network (Weber and Cheng, 2004). Unlike *one-to-one unicast*, *one-to-all broadcast*, or *one-to-many multicast*, anycast adopts the *one-to-one-of-many* delivery. A sender queries or sends data to an unspecified receiver, whose forwards such data to other receivers in the anycast group.

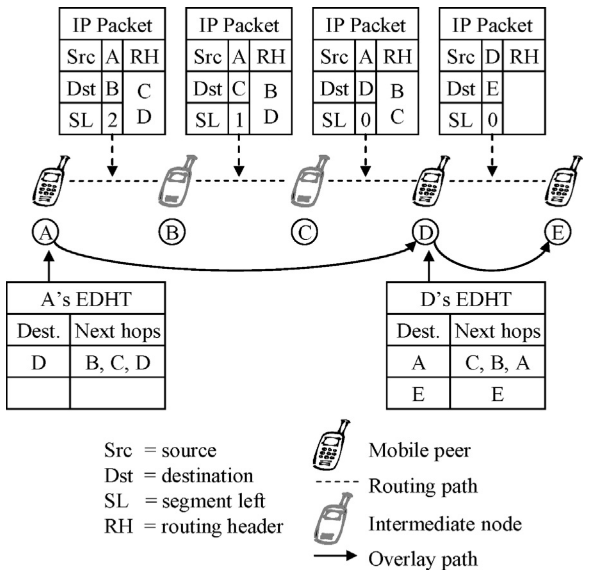


Fig. 1. The EDHT over IPv6 routing on COME-P2P.

Anycast originates from IPv6 for service-orientated applications to decrease network traffic and shorten response delay. An anycast address can be assigned in an anycast group, in which the receivers with the same anycast address should receive the same packets. Although the source should send a query to the nearest destination among an anycast group of multiple receivers, the nearest destination is not constant via different routing principles or arbitrary routing paths. Therefore, anycast is suited to connectionless protocols generally built under UDP.

3. Our proposed scheme

CACA is a cross-layer middleware under application layer and above MAC layer. The middleware design allows P2P file sharing or live streaming services on WiFi or WiMax network. The proposed scheme contributes to the multiple real-time P2P applications on mobile network, the overlay proximity and efficient routing, and the on-demand self-organization with scalability, recoverability, mobility, adaptability, extensibility, and interoperability.

3.1. System overview

As Fig. 2 illustrated, CACA is a cross-layer middleware of overlay layer, transport layer, and network layer.

- IPv6 for network layer: Anycast is used for service querying, and the extended routing header is used for hop-by-hop routing. Therefore, CACA uses IPv6 as the network layer protocol.
- UDP for transport layer: UDP can use a datagram socket to establish end-to-end communication in MANET. Its connectionless characteristics including of lightweight and datagram without acknowledgment, retransmission, and timeout, are suitable for wireless P2P real-time service. Importantly, UDP is suitable for IPv6 anycast and mobile hop-by-hop routing.
- P2P for overlay layer: P2P overlay is used to identify, index, and manage mobile nodes. The proposed *finger plus routing table* (*FPRT*) is used to construct the P2P overlay and support multiple services.

The FPRT is used to construct the P2P overlay. FPRT is based on EDHT to inherit the path information, but FPRT is shared as the combination of finger table and routing table on overlay tier and routing tier respectively in CACA. Path information is used for UDP and IPv6 routing to derive a local path between two peers through multiple intermediate nodes. An intermediate node without CACA still uses its routing table and ignores FPRT. Every mobile peer maintains its FPRT to know its members via the reactive notification of arrival or departure and the proactive detection of movement, so the path information is updated to accommodate the peers' mobility and churn. As a result, CACA achieves the high overlay proximity.

The high overlay proximity improves the high routing scalability, so the path information is used in a routing protocol in the proposed cross-layer scheme. CACA avoids the flooding scheme of routing discovery, and it uses the extended routing header in IPv6 to forward packets hop by hop, so the ad hoc routing protocol is replaced by the membership of overlay, such that the overhead is little and the packet collision is avoidable. Because the attribute next hop of IPv6 is based on the path information of FPRT, the overlay proximity leads to the routing efficiency.

3.2. Cross-layer scheme

CACA uses the cross-layer messages to implement the proposed middleware. When overlay tier or routing tier has actions or events, FPRT is informed to update for overlay proximity and routing efficiency. Because FPRT is shared in overlay tier and routing tier, the actions must be considered as Fig. 3 illustrated. For example, the arrival or departure of peers may lead to the redirection of routing path, which must change the overlay; therefore, the FPRT must be updated for the peer churn to achieve the overlay stability and scalability.

In CACA, FPRT must cache P2P service type and file query to support multiple P2P applications. When a peer queries one P2P service, anycast resolution is triggered to search on MANET, and then the results of query is cached in the intermediate peers. Because each mobile peer is similar as the router on a local wireless network, the service priority and anycast addressing are enabled to route packets possibly. A group of peers sharing the same content like as a *swarm* shares the same IPv6 anycast address.

Besides the reactive notifications of arrival and departure, each peer periodically monitors the neighbors to detect the movement or failure proactively. The network layer essentially broadcasts *Hello Message* periodically to keep alive, so the accumulation of *Hello Message* is used as a metric of peer mobility to update FPRT for overlay and routing optimization. Because *Hello Message* is provided essentially in MANET, the cross-layer approach does not increase additional overhead.

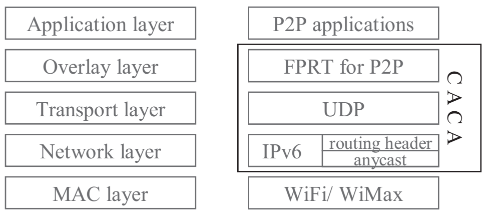


Fig. 2. The proposed cross-layer scheme (CACA).

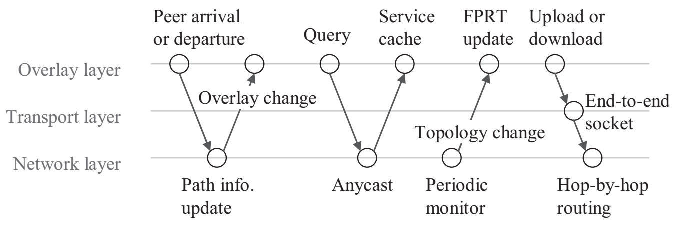


Fig. 3. The FPRT update between overlay tier and routing tier.

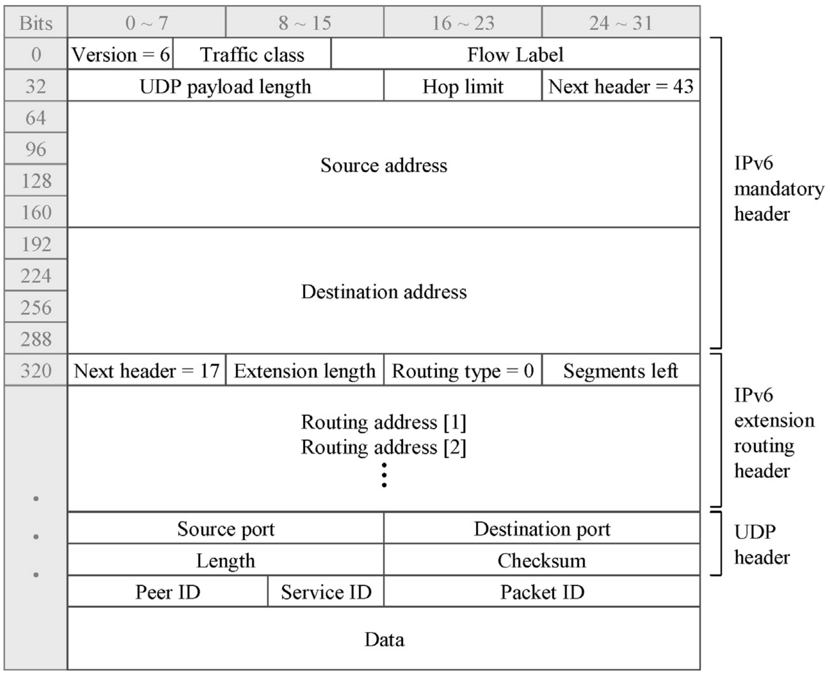


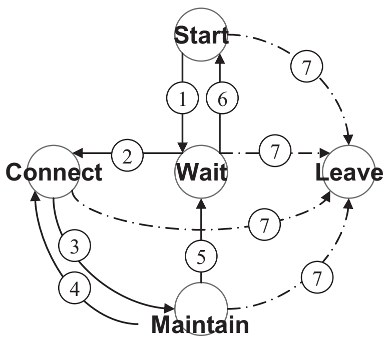
Fig. 4. The pseudo packet structure of CACA.

When delivering content including of file or multimedia stream, an end-to-end UDP socket is created to support a hop-by-hop routing path. The solution is described as Fig. 1 illustrated. The inter-layer packet is replaced with the proposed combination of packet headers as Fig. 4 illustrated. Because the packet format is based on UDP/IP, and the cross-layer approach is workable without any modification.

3.3. CACA design

At beginning, a peer joins in P2P network, and then it maintains its neighborhood overlay via exchanging signaling messages with other peers until leaving from P2P network. The lifecycle of peer represents the self-organization of overlay, and the final state machine is driven via such messages to design for the lifecycle and behaviors. As Fig. 5 illustrated, every peer should go through the states: start, wait, connect, maintain, and leave. Every current state should deal with the exact messages to avoid the incorrect errors, which crash overlay.

Based on Fig. 5, we give a normal example as in Fig. 6. Peer A is a new joining peer in the P2P network, and the joining process and querying process are described in the following:



① send *Join Message*
② receive *Join Accept Message*
③ receive *Join Message*
receive *Rejoin Message*
receive *Leave Message*
receive *Maintain Message*
timeout for *Hello Message*
④ send *Join Accept Message*
send *Join Reject Message*
send *Remain Message*
⑤ send *Rejoin Message*
⑥ receive *Join Reject Message*
timeout for *Join Message*
timeout for *Rejoin Message*
⑦ send *Leave Message*

Fig. 5. The final state machine or lifecycle of peer.

(1) First, because peer A neither knows anyone nor the overlay, it broadcasts *Arrival Message* to search the neighbors. The *Arrival Message* has the source address and routing trace.
(2) If an intermediate node receives *Arrival Message*, it does not know such message and rebroadcasts to search some peers. When a peer receives *Arrival Message*, it sends *Welcome Message* to peer A via the source address and routing trace, and it stops rebroadcasting. The *Welcome Message* has the destination address and routing trace.
(3) After receiving *Welcome Message*, peer A unicasts *Join Message* to select its member.
(4) After receiving *Join Accept Message*, peer A becomes one member in P2P network and can use P2P service.
(5) Peer A creates a *download task* when querying a file sharing or live streaming. If Peer A needs multiple services, it creates multiple download tasks.
(6) For example, peer A anycasts *Query Message* for a document.
(7) When receiving *Query Message*, the peer creates an *upload task* for a query.
(8) *Reply Message* with the queried result (i.e. yes or no) is sent to peer A. The queried result is informed to user.
(9) Peer A unicasts *Request Message* for the available document depending on the queried result.
(10) After receiving *Response Message*, peer A can download the queried document.
(11) The data is delivered until the download task is finished.
(12) Peer A leaves from the P2P network via *Leave Message*.

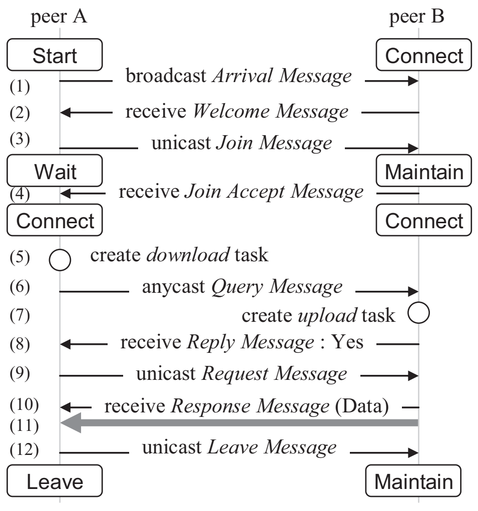

peer A
peer B
Start
Connect
(1) broadcast Arrival Message
(2) receive Welcome Message
(3) unicast Join Message
Wait
Maintain
(4) receive Join Accept Message
Connect
Connect
(5) create download task
(6) anycast Query Message
(7) create upload task
(8) receive Reply Message : Yes
(9) unicast Request Message
(10) receive Response Message (Data)
(11)
(12) unicast Leave Message
Leave
Maintain

Fig. 6. The normal message flow of CACA.

However, CACA process is not always normal during the overlay construction or data distribution. Therefore, we give an abnormal example based on Fig. 6 as Fig. 7 illustrated.

(13) When peer A choices the neighbor, peer B, as its member, it sends *Join Message* to peer B and expects to receive *Join Accept Message*. If *Join Message* or *Join Accept Message* is lost, the timeout of new joining process avoids the limitless waiting and is sent *Join Message* again.
(14) When receiving *Reply Message* with the queried result answering NO, peer A gives up the query.
(15) After peer B leaves, peer A searches other uploaders if peer B is only one uploader. Therefore, the download and upload task must have the final state machine to proceed the process.

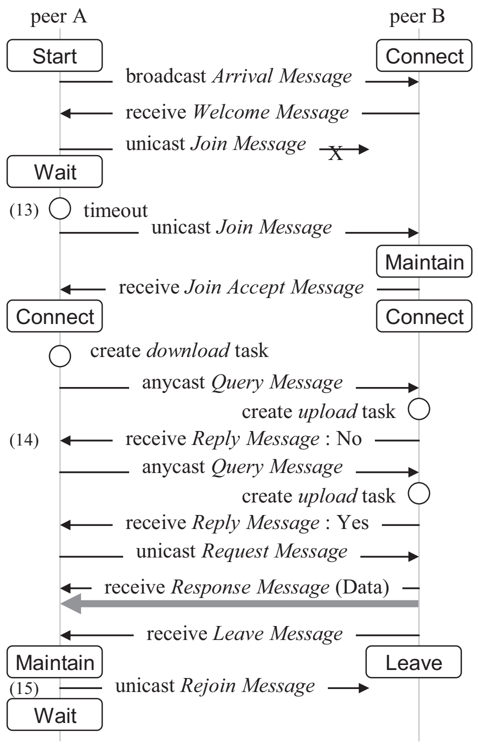
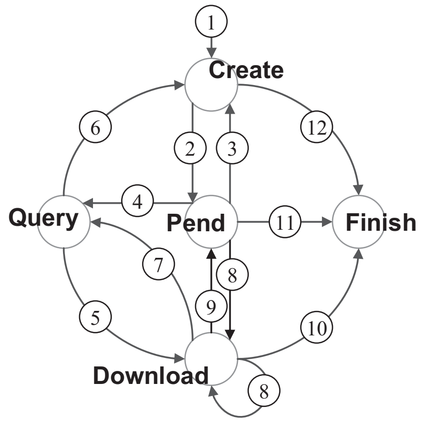


Fig. 7. The abnormal message flow of CACA.



①	demand query	⑦	timeout for *Request Message*
②	send *Query Message*	⑧	receive *Response Message* download data
③	timeout for *Query Message*	⑨	timeout for download peer adaptation
④	receive *Reply Message* : Yes or No	⑩	complete
⑤	*Reply Message*: Yes send *Request Message*	⑪	abort
⑥	*Reply Message*: No	⑫	cancel

Fig. 8. The final state machine or lifecycle of download task.

As Fig. 8 illustrated, CACA creates a download task when a demand query arrivals. The download task starts a timer when sending *Query Message*, and stops it when receiving *Reply Message*. The file or stream is shared via *Request Message* depending on *Reply Message*, and a timer is switched to proceed the download process. When the quality of download efficiency is low or the uploader is gone, CACA executes the peer reselection to search the other appropriate uploaders and waits for new *Reply Message*. The download task is completed to finish in a normal case, but it is aborted when pending the reply, or canceled when creating the query.

The download task should go through the states: create, pend, query, download, and finish. The upload task is operated in coordination with the download task, therefore, the upload task is created when receiving *Reply Message* and finished when receiving *Complete Message*. As Fig. 9 illustrated, the upload task should go through the states: create, reply, upload, and finish. Because the upload process is coordinated with the download process, we do not describe the superfluous detail.

The relation between peers is many-to-many, and the relation between upload and download tasks is many-to-many, too. Therefore, the message flow of CACA should combine the final state machines of peer behaviors and upload/download operations as Fig. 10 illustrated. In the scenario, peer A and D are new joining peers, and they download the same video stream from peer B and C individually. Let peer A plays the role acted via verbs in such scenario. During live streaming delivery, peer C leaves, and then peer D requests another member, i.e. peer B, for continuous streaming recovery. When peer A stops the streaming delivery, it sends *Complete Message* to peer B to finish the upload task. The upload/download operations of file sharing application are equal to the ones of live streaming application.

When a peer moves or leaves, the overlay must be adjusted to avoid far routing problem and service interruption. We give an example to explain the overlay maintenance and recovery of CACA as Fig. 11 illustrated.

(1) As Fig. 11(a) illustrated, the path A → C → E is connected, and all nodes (A, B, C, D, E) broadcast *Hello Message* to keep alive periodically on ad hoc protocols.

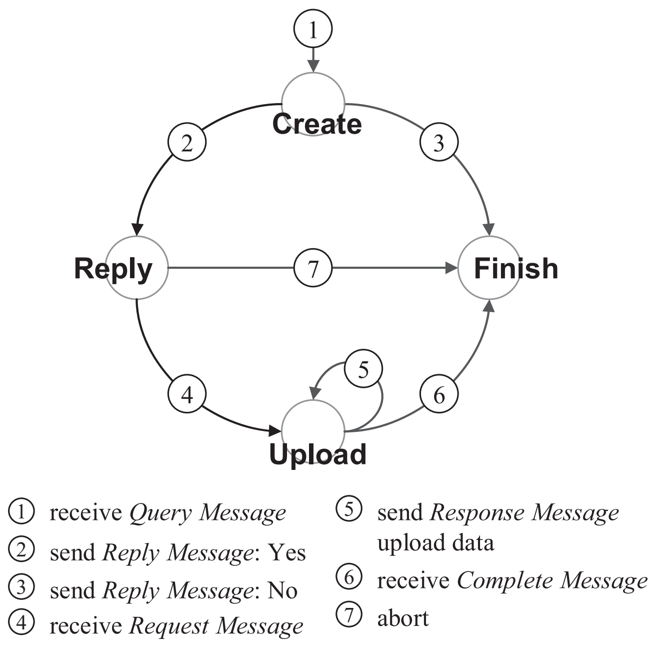

Create
Reply
Finish
Upload
1
2
3
4
5
6
7
① receive *Query Message*
② send *Reply Message*: Yes
③ send *Reply Message*: No
④ receive *Request Message*
⑤ send *Response Message* upload data
⑥ receive *Complete Message*
⑦ abort

Fig. 9. The final state machine or lifecycle of upload task.

(2) As Fig. 11(b) illustrated, after peer C moves, the accumulation of *Hello Message* from peer C to A decreases.
(3) When peer A considers that peer C moves far away, it sends *Maintain Message* to all peers on the path (only peer E in the example).
(4) Peer C is still routable via node D, so both peers A and E update the path information. One path becomes two paths A → C and A → E. Peer E returns *Remain Message* to avoid the interruption.
(5) If peer C cannot receive *Hello Message* from peers A and E, it sends *Rejoin Message* to peer A. If peer C receive data from peer A (via node D) before sending *Rejoin Message*, the step is ignored.
(6) Because peer A can route to peer C, it returns *Join Accept Message*, and then peer C recovers the overlay. If step (5) is ignored, the step is ignored, too.
(7) When peer C accumulates *Hello Message* from peers E, it updates the path information because peer E has been known.
(8) Peer C sends *Maintain Message* to peer A for the modification of path A → C, and peer C does to peer E for path A → C → E.
(9) Both peers A and E update the path information for path A → E → C and return *Remain Message* to peer C for the modification.
(10) As Fig. 11(c) illustrated, peer E sends *Leave Message* to all peers on the path before leaving from the P2P network. Peers A and C update the path information for path A → C as similarity of Figs. 6 and 7 illustrated.

CACA can manage P2P overlay on MANET to provide file sharing or live streaming service due to the cross-layer scheme and FPRT

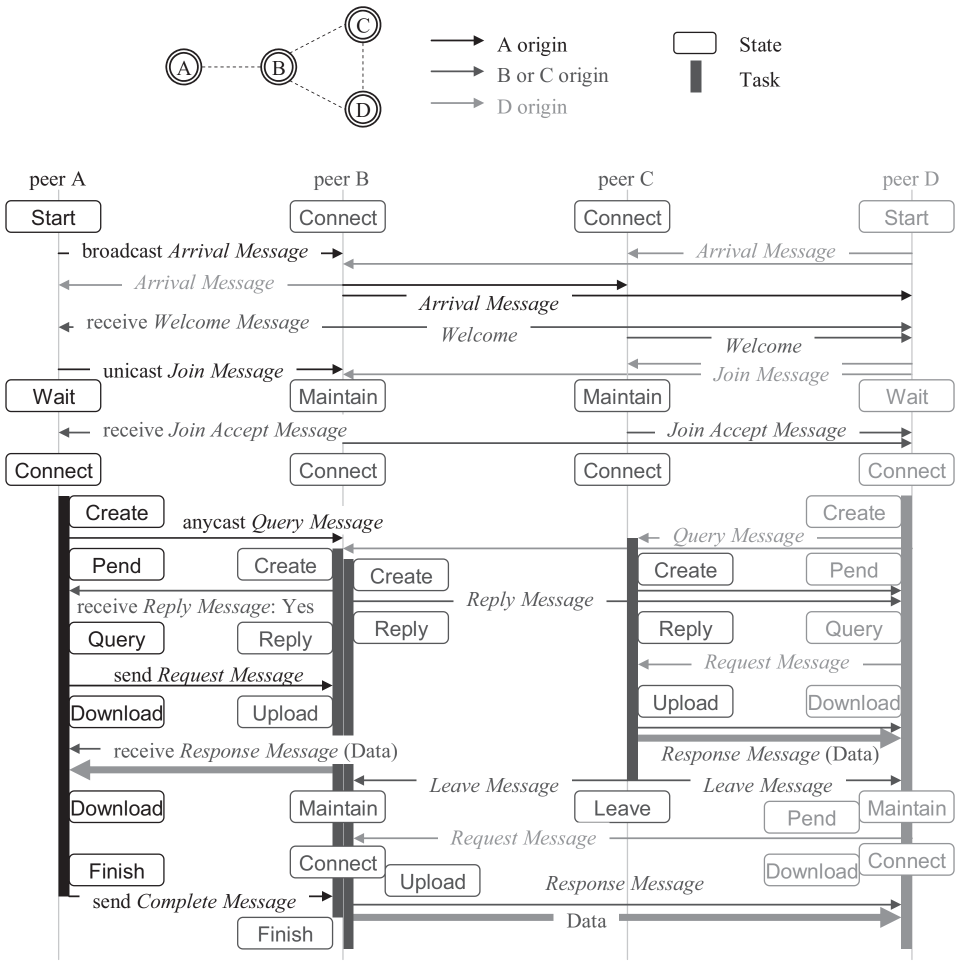


Fig. 10. The message flow of CACA.

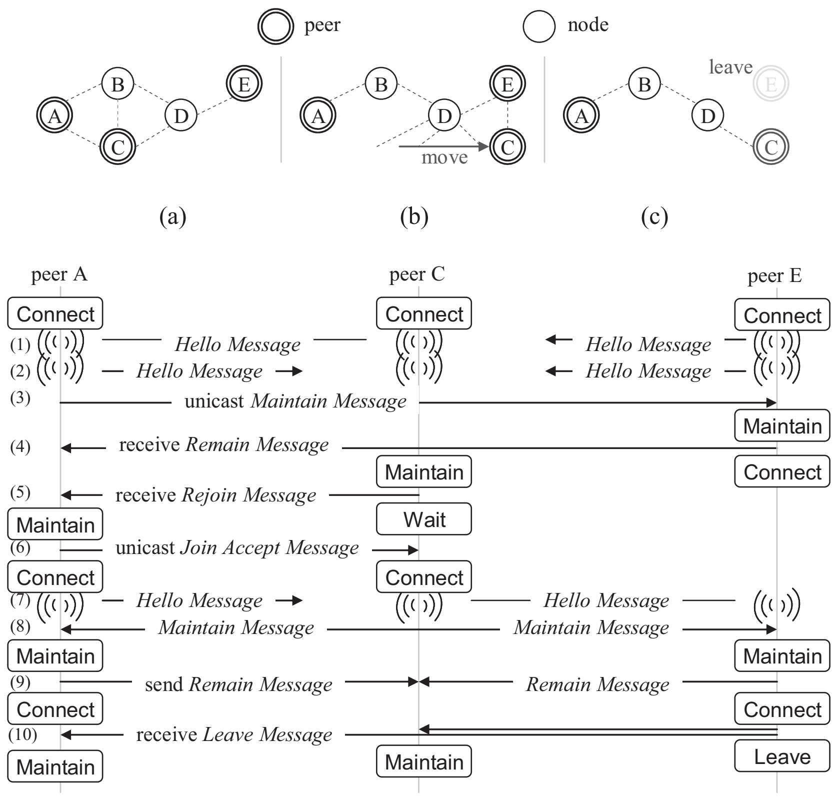


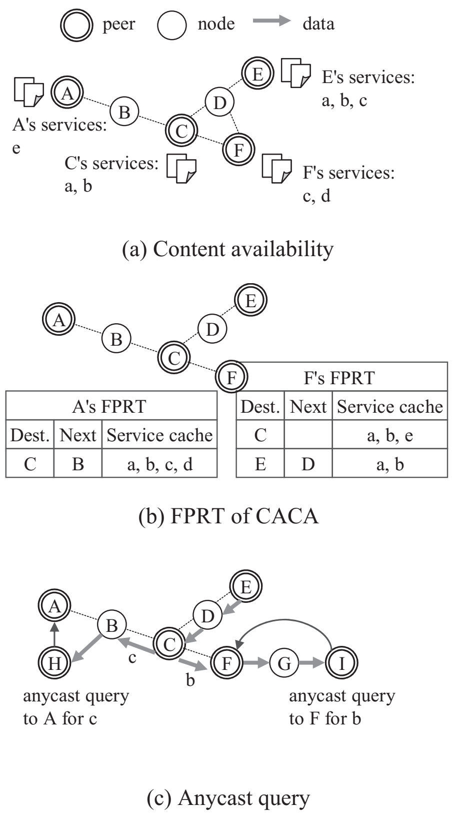
Fig. 11. The overlay maintenance and recovery of CACA.

design. Recall Fig. 3, even if the peers move or leave, FPRT is updated via the path information and service cache to keep the continuous data. The algorithms of CACA extend the P2P service, reduce the signaling overhead, and improve the recovery efficiency.

3.4. Algorithm

FPRT caches the service type, so CACA can accomplish the anycst query to extend all P2P services, such as file sharing, voice communicating, and video streaming. As Fig. 12 illustrated, there are peers A, C, E, and F in a local anycast group. As Fig. 12(a) illustrated, every peer owns the local services, e.g. peer A owns the service e, which may be a file, or peers C and E owns the services a and b, which may be a voice conference. As Fig. 12(b) illustrated, every peer updates its FPRT via querying on demand. If peers cannot communicate within one hop, the intermediate nodes as B and D relay to establish a P2P connection. The connection maintenance is updated in the path information, which is combined with the service cache in FPRT. Although peer A owns only service e, it makes services a, b, c, and d available via peer C on P2P overlay.

As Fig. 12(c) illustrated, peers H and I start CACA to join in such P2P network. Peer H creates a download task to query for service c, and then if peer A receives such *Query Message*, peer A returns a *Reply Message* with the queried result (YES). Because peer H is routable to peer C, the routing path E → D → C → B → H can be derived. On the other hand, peer I creates a download task to query for service b, and then if peer F receives such *Query Message*, peer F returns a *Reply Message* with the queried result (YES). Because peer I is unroutable to peers C and E, the routing path C → F → G → I can be derived. Of course, the intermediate nodes do not know CACA, so they just forward the messages via the ad hoc routing protocol. The reactions of *Query Message* and *Reply Message* are described as Fig. 13 illustrated.



A's FPRT

Dest.	Next	Service cache
C	B	a, b, c, d

F's FPRT

Dest.	Next	Service cache
C		a, b, e
E	D	a, b

Fig. 12. FPRT process to service cache and query.

```
when a Query Message is received do
    let QM be such message;
    let RM be a Reply Message;
    if QM.getPeerId( ) ∈ this.members{ } then
        new a UT to create a upload task;
        UT.setState(Create);
        UT.setPeerId(QM.getPeerId( )) ;
        if QM.getQueryService( ) ∈ this.FPRT.cache{ } then
            RM.setResult(true);
            RM.setCache(this.FPRT.cache);
            UT.setState(Reply);
        else
            RM.setResult(false);
            UT.setState(Finish);
        RM.setQueryService(QM.getQueryService());
        sends RM to QM.getSource( );
    else
        drop such message;
end when


when a Reply Message is received do
    let RM be such message;
    if RM.getQueryService()  ∈  this.download{ } then
        this.download();
    else
        drop such message;
end when


when download task receives a Reply Message do
    let RM be such message;
    if getState( ) is Pend then
        setState(Query);
        .
        .
        .
        if RM.getResult( ) is true
            setState(Download);
            .
            .
            .
        else
            setState(Create);
            .
            .
            .
    else
        drop such message;
end when
```

Fig. 13. The proposed algorithms of query process.

4. Analysis

Via the complexity analysis of theoretical computation, the *routing complexity* and *maintenance load* are used to analyze the efficiency of compared schemes.

4.1. Routing complexity

In general, one P2P scheme combined with one ad hoc routing protocol provides the P2P service on MANET. For example, Ad-hoc On-demand Distance Vector (AODV) (Perkins and Royer, 1999) is a famous hop-by-hop routing protocol, and it always establishes the P2P communication on demand. P2P solution can search data via flooding or DHT query. The flooding query is suitable for dynamic topology, but the DHT query is efficient. We compare CACA with other schemes including of the flood-querying and DHT-querying over AODV.

Let n be the number of mobile peers. In flood-querying over AODV, a peer searches a source via flooding query, whose complexity is well-known O(n). Every query is forwarded on demand by intermediate node via AODV, whose complexity is well-known O($\log n$). Therefore, the routing complexity of flooding over AODV is O($n \log n$).

In DHT-querying over AODV, a peer searches a source via DHT-based approach, whose complexity is well-known O($\log n$). The complexity of AODV is well-known O($\log n$).

Table 1
A summary of P2P over MANET schemes.

	Routing complexity	Maintenance load
Flood-querying over AODV	$O(n \log n)$	$O(n)$
DHT-querying over AODV	$O((\log n)^2)$	$O((\log n)^2)$
The proposed cross-layer	$O(\log n)$	$O(\log n)$

Therefore, the routing complexity of DHT-querying over AODV is $O((\log n)^2)$.

In CACA, a peer searches a source via FPRT, which inherits EDHT, thus its complexity is $O(\log n)$. The routing path is derived via FPRT to avoid the choice of on-demand intermediate node. The complexity of IPv6 routing via hop-by-hop routing header is $O(1)$. Therefore, the routing complexity of proposed scheme is $O(\log n)$.

4.2. Maintenance load

Maintenance load means that each peer requires the maintenance load of P2P overlay when overlay is changed.

In flood-querying over AODV, a peer usually discovers its members for reactive querying, so the proactive overlay maintenance is unnecessary. Each peer just checks the arrival or departure of its member during the P2P communication, and the load is $O(1)$ trivially. In addition, the maintenance load of AODV is well-known $O(n)$ due to rediscovery. Therefore, the maintenance load of flooding over AODV is $O(n)$.

In DHT-querying over AODV, a peer maintains its DHT for indexing peers and files, and the maintenance load of DHT is well-known $O(\log n)$. AODV reestablishes a P2P communication without rediscovery due to the proactive maintenance of DHT, and the reestablishment of hop-by-hop path is well-known $O(\log n)$. Therefore, the maintenance load of DHT-querying over AODV is $O((\log n)^2)$.

In CACA, a peer maintains its FPRT for path information and service cache. The both updates of path information and service cache cost $O(\log n)$. Because FPRT shares the finger table and routing table, the maintenance of ad hoc routing is $O(1)$. Therefore, the maintenance load of CACA is $O(\log n)$. In summary, the routing complexity and maintenance load are summarized as Table 1 illustrated.

4.3. Compared schemes

File sharing and live streaming are the common applications of P2P over MANET. We summarize the compared schemes as Table 2 illustrated, and list the descriptions.

- ORION (Optimized Routing Independent Overlay Network) (Klemm et al., 2003) is based on an application layer overlay combined with the reactive ad hoc routing protocol for file sharing to guarantees high data rates and low transmission overhead over MANET. ORION integrates Gnutella (Ripeanu, 2001) query processing and overlay network construction with AODV routing discovery. Gnutella is a flooding base, so ORION belongs to the flood-querying over AODV.
- MADPastry (Mobile AD hoc Pastry) (Zahn and Schiller, 2005) is a file sharing application, and it belongs to the DHT-querying over AODV. MADPastry uses Pastry (Rowstron and Druschel, 2001) proximity awareness to reduce the overhead without flooding. Pastry routing table indexes and hashes the mobile nodes
- Smart Gnutella (Boukerche et al., 2006) enhances the original Gnutella to suit for MANET and for the real-time application. Because the periodic *ping/pong* and *broadcast* can assist in routing maintenance, Smart Gnutella adopts AODV as the ad hoc routing protocol. Therefore, Smart Gnutella belongs to the flood-querying over AODV.
- MP2PS (Mesh-based P2P Streaming) (Qadri et al., 2008) is a live streaming application, and it belongs to the DHT-querying over AODV. MP2PS provides real-time streaming with scalability and availability over MANET. MP2PS adopts mesh-based live streaming application, Joost (Alhaisoni and Liotta, 2009), and no retransmission on wireless network, UDP, and on-demand ad hoc routing protocol, AODV.

Table 2
The comparison of schemes.

	File sharing	Live streaming
Flood-querying over AODV	ORION	Smart Gnutella
DHT-querying over AODV	MADPastry	MP2PS

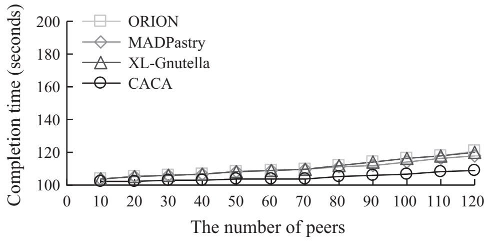


Fig. 14. Download completion time vs. the number of nodes.

In addition, we also compare CACA with XL-Gnutella (Cross-Layer Gnutella) (Conti et al., 2005) to discuss the effect of ad hoc routing protocol. XL-Gnutella is based on Gnutella overlay and OLSR (Clausen et al., 2001) ad hoc routing protocol. Peer discovery refreshes overlay maintenance in Gnutella's flooding query, and link selection prioritizes the close connections, with the goal of building an overlay network topologically proximal to the physical network.

5. Results

We use OMNet++ 4.0 to simulate the P2P service on WiFi, and use INET to simulate the network behavior including organizing an ad hoc topology and handling messages. We use MF (Mobility Framework) module to simulate wireless MAC layer and use OverSim module to simulate P2P overlay layer. The simulation adopts RTS/CTS mechanism and random waypoint model. 120 mobile nodes move inside a bounded area of $900\,\text{m} \times 900\,\text{m}$, and the radio transmission range of a node is $125 \pm 25\,\text{m}$. Although the ideal bandwidth of WiFi standard is 11 Mbps, the estimated value of our measurement is less than 2 Mbps under ad hoc model in reality. As a result, we set the bandwidth 1000 ± 200 kbps without any fading interference. Every statistic value of simulation result is the average of 30 repeated experiments.

5.1. File sharing

We characterize the download strategy and file factors on BitTorrent (Johnsen et al., 2005) to build a simulated benchmark. The total file size is 8 MB, the file piece size is 256 kb, the packet size is 512 bytes, and the data rate is best effort. The *download completion time* is a major metric of file sharing, and it means how much time a file is finished downloading. We evaluate it via the network size, moving speed, and churn rate in a many-to-many delivery scenario.

As Fig. 14 illustrated, the proposed CACA has the shortest completion time among the compared schemes. Because CACA and MADPastry inherit the DHT query, the querying latency of CACA and MADPastry is shorter than that of ORION and XL-Gnutella.

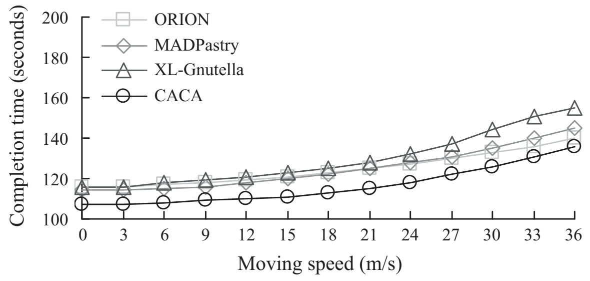


Fig. 15. Download completion time vs. the moving speed.

CACA uses not only DHT-based index but also anycast query, so the querying latency of CACA is shorter than that of MADPastry, ORION, and XL-Gnutella. After finishing query or constructing overlay, the AODV routing discovery or connection must be proceeded in MADPastry and ORION. However, the routing path can be derived from FPRT due to the integration of finger and routing table in CACA, so CACA shortens the completion time. The completion time is lengthened gracefully with the increasing network size, so the proposed CACA is demonstrated with scalability.

As Fig. 15 illustrated, although the completion time is lengthened with the increasing moving speed, the service is still workable, so the proposed CACA is demonstrated with mobility. CACA can detect the peer movement via the accumulation of *Hello Message* to update path information in FPRT due to the cross-layer scheme. The update of path information not only recovers the routing path but also improves the overlay proximity, which avoids the far routing problem. Although the peers periodically probe their neighbors in MADPastry and ORION, the far routing problem cannot be avoidable. The routing path may be disconnected, and AODV reestablishes the routing discovery, which lengthens the service interruption latency. Although cross-layer event is triggered in XL-Gnutella, OLSR does not inform Gnutella the peer movement. The overlay maintenance and data cache degrade the efficiency of Gnutella, because OLSR updates routing table frequently.

The peer arrival/departure must change the P2P overlay, and we define that the *churn rate* is the probability of peer arrival/departure. Given a random interval, every peer stays or leaves depending on the churn rate. As Fig. 16 illustrated, the proposed CACA has the shortest completion time among the compared schemes. Because CACA and MADPastry inherit the DHT maintenance, every peer can maintain its members to recover a broken overlay. Every peer builds the local overlay on demand in ORION, so the overlay is suitable for a dynamic environment. Both CACA and XL-Gnutella use cross-layer design to recover P2P overlay, and update routing table directly. The peer adaptation or link selection can be executed immediately due to the proximity. Of course, the parts of file become unavailable because the source leaves, so the completion time become much long or the file cannot be downloaded completely when the churn rate is high. When the churn rate is 0.5, the completion time is 148 s in CACA, so the proposed CACA is demonstrated with recoverability.

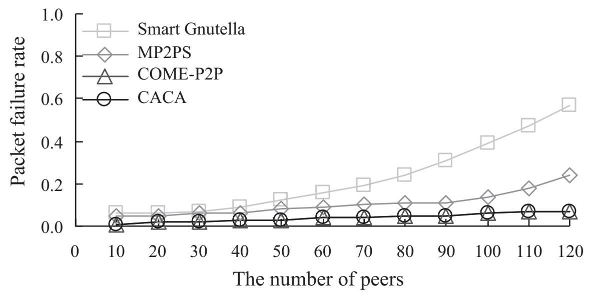


Fig. 17. Packet failure rate vs. the number of peers.

5.2. Live streaming

In order to build a simulated benchmark, we characterize the delivery principle, the overlay construction, and the streaming factors on Joost (Lei et al., 2010) referred to MP2PS. The video packet size is 1024 bytes, and the data rate of streaming is 450 kbps (constant bit rate). The *packet failure rate* is a major metric of live streaming, and it is defined as the number of video chunks that lose or arrive before playback deadlines over the total number of video chunks. In the experiment, we consider a one-to-many delivery scenario.

As Fig. 17 illustrated, the proposed CACA has the lowest packet failure rate among the compared schemes. Because CACA is an extension of COME-P2P for live streaming delivery, the performance of CACA is similar with that of COME-P2P. When the number of nodes increases to 120, the failure rate is 0.07, so the proposed CACA is demonstrated with scalability. However, 0.07 of failure rate is still hardly considered a good result in terms of video transmission given the high level of interdependency of bitstreams. We find that there are traffic jitters in the transmission, and the traffic jitters result from wireless random access and hidden terminal problem, and lead to unstable streams. We will consider the buffering mechanism to keep the playback continuity.

As regards the compared schemes, MP2PS cannot support high scalability because the overlay forwarding is not mapped into the ad hoc routing. The larger number of peers is relative to the more routing paths, so the flooding query degrades the streaming continuity in Smart Gnutella. There are also many traffic jitters in the transmission in MP2PS and Smart Gnutella.

In wireless network, the mobility problem leads to a difficulty of keeping a continuous playback. The number of mobile nodes is 100 in the simulation for mobility. As Fig. 18 illustrated, the cross-layer design can be appropriate for the mobility. Due to the overlay maintenance and overlay proximity of CACA, it can maintain a low failure rate and low overhead when mobile peers move. When moving speed is 36 m/s, the packet failure rate is lower than 0.4. There are average 3.25 hops in a peer-to-peer path, and there are average 22.35 nodes in a streaming path. Smart Gnutella also has stable performance against the moving speed, because the moving

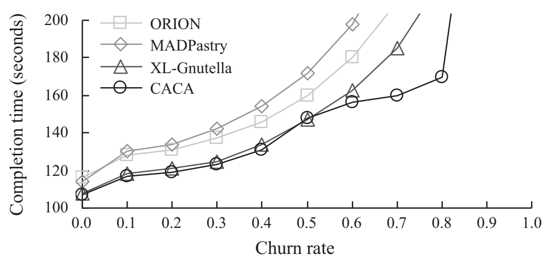


Fig. 16. Download completion time vs. the churn rate.

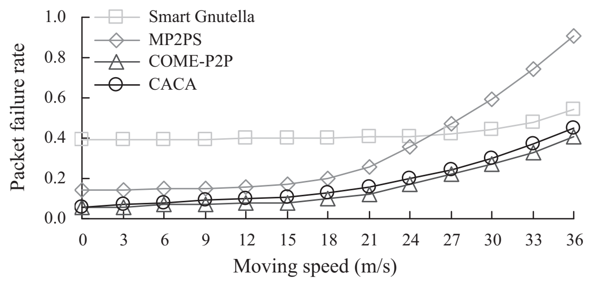


Fig. 18. Packet failure rate vs. the moving speed.

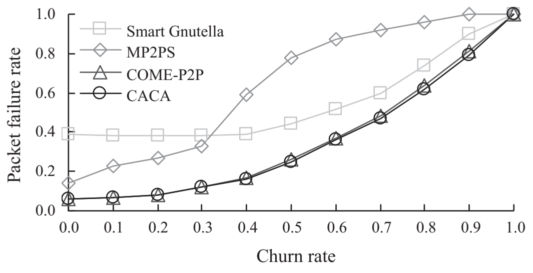


Fig. 19. Packet failure rate vs. the churn rate.

speed affects the flooding scheme slightly. However, DHT originally is designed for wired network, so MP2PS is unable to detect peer movement, such that the peer cannot find its nearest neighbor to forward stream. Under the high moving speed, CACA highlights the advantage for mobility.

For live streaming, the peer churn leads to a difficulty of keeping a stable overlay. As Fig. 19 illustrated, the proposed CACA is suitable for a dynamic MANET, because the cross-layer design speeds up recovery latency and improves streaming stability. FPRT is always updated to recover backup routing path, but ad hoc routing disconnection or rediscovery on AODV may be conflicted with overlay recovery in Smart Gnutella and MP2PS. Therefore, the packet failure rate is only 0.25 in CACA when the churn rate is 0.5, however, that is 0.44 and 0.78 in Smart Gnutella and MP2PS, respectively. CACA is suitable for dynamic network with high churn rate.

5.3. Signaling overhead

The *signaling overhead* includes routing overhead and overlay overhead, so we define the signaling overhead as the number of non-data packets sent per peer per second. As Fig. 20 illustrated, the proposed CACA has the lowest signaling overhead among the compared schemes, and we discover several observations.

- In Smart Gnutella, the major overhead is routing (especially querying) overhead, because the on-demand flooding query leads to high overhead. The signaling overhead increases linearly with network size, and it lacks scalability. On the other hand, the signaling overhead of DHT-based schemes (such as MP2PS, COME-P2P, CACA) is stable with scalability.
- In MP2PS, the major overhead is routing overhead, because the routing discovery needs an additional overhead to plot the DHT-based forwarding.
- In COME-P2P, the major overhead is overlay overhead, because the periodical estimation is needed to measure the latency, detect peer churn and movement.
- Although CACA has the similar playback continuity to COME-P2P as Fig. 17 illustrated, CACA reduces much signaling overhead, because CACA detects peer churn and movement via the accumulation of *Hello Message*, which need not any modification or overhead. The proposed anycast query also reduces the signaling overhead, because anycast minimizes the number of querying hops.

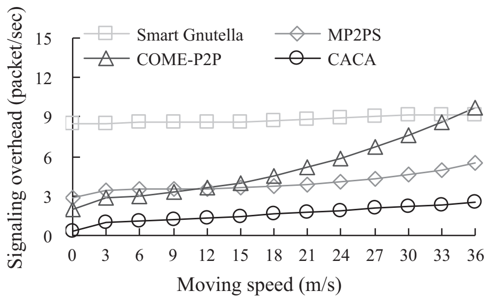


Fig. 21. Signaling overhead vs. the moving speed.

As Fig. 21 illustrated, the signaling overhead of CACA is less than that of COME-P2P, because FPRT is updated without additional measurement. However, CACA sometimes mistakes the peer movement for peer churn, so the packet failure rate of CACA is higher than that of COME-P2P as Fig. 18 illustrated. On the other hand, in Smart Gnutella, because a peer searches its members when querying real-time data on demand, the moving speed affects the signaling overhead slightly.

As Fig. 22 illustrated, the signaling overhead of CACA linearly increases when churn rate rises. In the proposed scheme, when a peer joins, its neighbors inform each other to update their FPRTs and derive the routing path; and when a peer leaves, it proactively informs its neighbors. As a result, CACA generates the high signaling overhead at high churn rate. COME-P2P has similar overhead due to the similar cross-layer design, the periodical latency estimation also adds overhead. Similarly, MP2PS faces the difficulty of peer churn, but it is unable to derive routing path in advance to economize signaling overhead.

There are three disadvantages of Smart Gnutella and MP2PS: inefficient routing for live high-bit-rate streaming, high on-demand routing overhead, low mobility and far routing problem. The simulation results demonstrate that the cross-layer integration of FPRT and IPv6 can avoid the disadvantages. Both Smart Gnutella and MP2PS cannot change the logical link on overlay, so the schemes perform AODV to visit many unnecessary intermediate nodes.

Fast moving speed and high churn rate cannot produce much overhead to crash CACA. The proposed FPRT really maintains neighborhood and interacts cross layers for deriving routing path. Via integrated with anycast and routing header in IPv6 routing, the signaling overhead is less than 0.5% over total streaming traffic. The overlay proximity is suitable for dynamic MANET and improves routing efficiency.

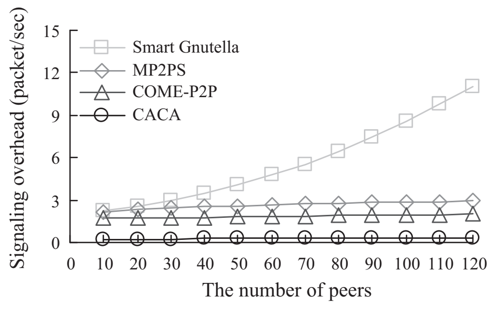


Fig. 20. Signaling overhead vs. the number of peers.

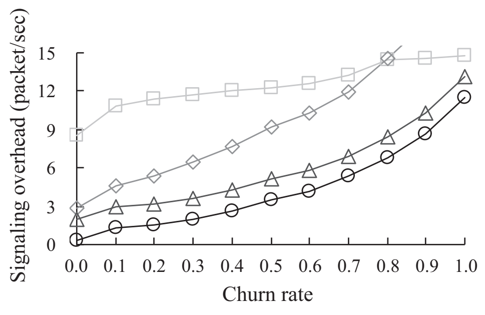


Fig. 22. Signaling overhead vs. the churn rate.

6. Conclusion

In this paper, we present a cross-layer design for P2P application in MANET. The proposed scheme, CACA (Context-Aware Cooperative Application) integrates the routing protocol with overlay protocol for all P2P services on the dynamic wireless network. A combination of the proposed FPRT (Finger Plus Routing Table), path information, and IPv6 routing protocol can manage neighboring peers and derive the routing path for reactive query or proactive delivery due to logical overlay proximal to physical topology. Therefore, the proposed scheme can provide the stable routing path to improve download delay, playback continuity, and low signaling overhead for high data rate and low latency in P2P service when facing scalability, mobility, churn with the reasonable overhead.

CACA can be seen as an extension of COME-P2P, and CACA uses a cross-layer middleware to integrate DHT-based lookup, anycast query, and P2P delivery via the IPv6 routing header. We demonstrate that CACA has outstanding performance on routing efficiency and traffic overhead for high bit-rate delivery via both the experimental discussion. Depending on the mathematical analysis, CACA has the logarithmical increase of complexity and overhead with scalability; depending on the simulation results, CACA has the best performance among the compared schemes.

As the discussion of simulation shown, CACA accommodates the large number of users for scalability, handles the high churn rate for recoverability, detects peer movement for mobility, integrates P2P with MANET for adaptability, supports the all kinds of P2P services for extensibility, and supplies different resources for interoperability. We will consider context elements such as user mobility profile, available battery, or available wireless networks to increase cross-layer interaction. Although this proposed scheme cannot be suitable for IPv4 network nowadays, an integration with mobile IPv6 will be an extension in the future.

References

Alhaisoni, M., Liotta, A., 2009. Characterization of signaling and traffic in Joost. Peer-to-Peer Networking and Applications 2 (1), 75–83.

Boukerche, A., Zarrad, A., Araujo, R.B., 2006. A smart Gnutella overlay formation for collaborative virtual environments over mobile ad-hoc networks. In: IEEE International Symposium on Distributed Simulation and Real-Time Applications.

Chou, C.C., Wei, D.S.L., Kuo, C.C.J., Naik, K., 2007. An efficient anonymous communication protocol for peer-to-peer applications over mobile ad-hoc networks. IEEE Journal on Selected Areas in Communications 25 (1), 192–203.

Clausen, T., Jacquet, P., Laouiti, A., Muhlethaler, P., Qayyum, A., Viennot, L., 2001. Optimized link state routing protocol for ad hoc networks. In: IEEE International Multitopic Conference.

Conti, M., Gregori, E., Turi, G., 2005. A cross-layer optimization of Gnutella for mobile ad hoc networks. In: ACM International Symposium on Mobile ad hoc Networking and Computing.

Haw, R., Hong, C.S., Kim, D.S., 2009. Group P2P network organization in mobile ad-hoc network. In: Asia-Pacific Network Operations and Management Symposium.

INET Framework. http://inet.omnetpp.org

Johnsen, J.A., Karlsen, L.E., Birkeland, S.S., 2005. Peer-to-Peer Networking With BitTorrent. Department of Telematics, NTNU.

Klemm, A., Lindemann, C., Waldhorst, O.P., 2003 Oct. A special-purpose peer-to-peer file sharing system for mobile ad hoc networks. In: IEEE Vehicular Technology Conference.

Kuo, J.L., Shih, C.H., Ho, C.Y., Chen, Y.C., 2012. A cross-layer approach for real-time multimedia streaming on wireless peer-to-peer ad hoc network. Ad hoc Networks 11 (1), 339–354.

Lei, J., Shi, L., Fu, X., 2010. An experimental analysis of Joost peer-to-peer VoD service. Peer-to-Peer Networking and Applications 3 (4), 351–362.

Li, M., Chen, E., Sheu, P.C., 2006. A chord-based novel mobile peer-to-peer file sharing protocol. In: Asia-Pacific Web Conference on Frontiers of WWW Research and Development.

Mobility Framework for OMNeT++. http://mobility-fw.sourceforge.net/hp/index.html

OMNET++ 4.0. http://www.omnetpp.org

OverSim for OMNeT++. http://www.oversim.org

Peng, G., Li, S., Jin, H., Ma, T., 2004. M-CAN: a lookup protocol for mobile peer-to-peer environment. In: International Symposium on Parallel Architectures, Algorithms and Networks.

Perkins, C., Royer, E., 1999. Ad hoc on-demand distance vector routing. In: IEEE Workshop on Mobile Computing Systems and Applications.

Qadri, N.N., Alhaisoni, M., Liotta, A., 2008. Mesh based P2P streaming over MANETs. In: International Conference on Advances in Mobile Computing and Multimedia.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S., 2001. A scalable content-addressable network. In: ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications.

Ripeanu, M., 2001. Peer-to-peer architecture case study: Gnutella network. In: International Conference on Peer-to-Peer Computing, August.

Rowstron, A., Druschel, P., 2001. Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms.

Sesay, S., Yang, Z., He, J., 2004. A survey on mobile ad hoc wireless network. Information Technology Journal 3 (2), 168–175.

Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H., 2001. Chord: a scalable peer-to-peer lookup service for Internet applications. In: ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications.

Weber, S., Cheng, L., 2004. A survey of any cast in IPv6 networks. IEEE Communications Magazine.

Zahn, T., Schiller, J., 2005. MADPastry: a DHT substrate for practicably sized MANETs. In: Workshop on Applications and Services in Wireless Networks.

Jun-Li Kuo works in Information and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan. He received his Ph.D. in Computer Science from National Chiao Tung University, Hsinchu, Taiwan in 2013. His research interests include peer-to-peer network, wireless multimedia, ad-hoc network, and clouding computing.

Chen-Hua Shih works in Chung Shan Institute of Science and Technology, Taoyuan, Taiwan. He received his Ph.D. in Computer Science from National Chiao Tung University, Hsinchu, Taiwan in 2012. His research interests include peer-to-peer network, QoS, mobile IP, wireless networks, and network protocols.

Cheng-Yuan Ho is a R&D manager in the Advanced Research Institute, Institute for Information Industry, Taipei, Taiwan. His research interests include the design, analysis, and modeling of the congestion control algorithms, mobile and wireless networks, high speed networking, P2P networks, real-flow test, and quality of service. Ho received his Ph.D. in Computer Science from National Chiao Tung University, Hsinchu, Taiwan in 2008.

Yaw-Chung Chen received his Ph.D. degree from Northwestern University, Evanston, Illinois, USA. In 1986 he joined AT & T Bell Laboratories, where he worked on various exploratory projects. He is currently a professor in the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan. His research interests include Internet Protocols, fast mobility management, P2P systems and green computing.