Wireless Pers Commun (2007) 43:549–571
DOI 10.1007/s11277-007-9249-4

RAA: a ring-based address autoconfiguration protocol in mobile ad hoc networks

Yuh-Shyan Chen · Tsung-Hung Lin · Shih-Min Lin

Received: 15 May 2006 / Accepted: 1 January 2007 / Published online: 20 April 2007

Abstract The problem for dynamic IP address assignment is manifest in mobile ad hoc networks, especially in 4G all-IP-based heterogeneous networks. Existing solutions are mainly riveted to decentralized algorithms, applying a large number of broadcast messages to (1) maintain available IP address pools and (2) ensure no address duplication occurring. In this paper, we propose a ring-based address autoconfiguration protocol to configure node addresses. This work aims at the decentralized ring-based address autoconfiguration (DRAA) protocol, which has the advantage of low latency, low communication overhead and high uninterruptible connection. The DRAA protocol is a low-latency solution because each node independently allocates partial IP addresses and does not need to perform the duplicate addresses detection (DAD) during the node-join operation. Communication overhead is significantly lessened in that DRAA protocol uses the logical ring, thus utilizing fewer control messages solely by means of uni-cast messages to distribute address resources and to retrieve invalid addresses. Furthermore, if duplicate addresses are shown at network merging, the DRAA protocol checks the number of both TCP connections and of nodes to allow duplicate nodes to rejoin the smaller network so that lost connections are fast re-connected. To improve communication overhead and provide the evenness of address resources, the centralized ring-based address autoconfiguration (CRAA) protocol is discussed. The CRAA protocol reduces larger numbers of broadcast messages during network merging. The other contribution is that our CRAA protocol also has an even capability so that address resources can be evenly distributed in each node in networks; this accounts for the reason our solution is suitable for large-scale networks. Finally, the performance analysis illustrates performance achievements of RAA protocols. The simulation result shows that the DRAA protocol has the shortest latency, that the CRAA protocol has the capability to evenly distribute address resources, and that both of DRAA and CRAA protocols are the good solutions which achieve low communication overhead and high uninterruptible connection.

Keywords Autoconfiguration · IP address assignment · MANET · RAA · Wireless IP

Y.-S. Chen (✉) · S.-M. Lin
Department of Computer Science and Information Engineering, National Taipei University, Taipei County, Taiwan, Republic Of China
e-mail:yschen@csie.ntpu.edu.tw

T.-H. Lin
Department of Information Management, Hsing Wu College, Taipei County, Taiwan, Republic Of China
e-mail:duke@mail.hwc.edu.tw

1 Introduction

Multiple functions of the fourth-generation (4G) communication system are envisioned to be extensively used in the near future. 4G networks are an all-IP-based heterogeneous network, exploiting IP-based technologies to achieve integration among multiple access network systems, such as 4G core networks, 3G core networks, wireless local area networks (WLANs) and Manifest in mobile ad hoc networks (MANETs). In IP-based MANETs, users communicate with others without infrastructures and service charges. A MANET is made up of identical mobile nodes, each node with a limited wireless transmission range to communicate with neighboring nodes. In order to link nodes through more than one hop, multi-hop routing protocols—such as DSDV [9], AODV [10], DSR [5], FSR [3], ZRP [20] and OLSR [4]—are designed. These multi-hop routing protocols require each node to have its own unique IP address to transmit packets hop-by-hop toward the destination. Hence to practice these routing protocols in a correct manner, a node must possess an IP address bearing no similarity to that of any other node. Provided that two nodes possess the same IP address, then packets are likely to be transmitted toward a wrong direction. Therefore, the efficiency and accuracy of routing protocols depends on whether a node in MANETs is assigned with a unique IP address.

In hardwired networks, hosts generally employ the Dynamic Host Configuration Protocol (DHCP [2]) and IPv6 stateless address autoconfiguration [15] for dynamic IP address assignment. MANETs, by contrast, does not rely on a fixed server like DHCP because nodes in networks have joining, leaving, partitioning and merging behaviors. In the process of network merging, the duplicate address detection (DAD) plays an important role as the detector in IP address assignment. Because two or more nodes using the same IP address in a MANET inevitably cause packets to be routed toward the wrong destination, much research focuses on decentralized algorithms, enabling dynamic IP address assignment to function properly in MANETs. However, the DAD procedure in all existing solutions costs excessive communication overhead which consumes much bandwidth.

In recent years, many solutions for dynamic IP address assignment in MANETs have been brought out. According to their dynamic addressing mechanisms, we organize these solutions into the following four categories: all agreement approaches [8,11,18], leader-based approaches [13,17], best-effort approaches [16,19] and buddy system approaches [6,14]. In Table 1, four typical protocols from four different types of approaches are selected to compare with RAA protocols on characteristics and performances. The first row is the category of protocols. Their characteristic comparisons are shown from the third to seventh rows. The quantitative evaluation of selected protocols is presented in the last three rows. The notations are used to explain comparisons first: the node number in a MANET is n, the MANET diameter D, the average node degree d, the average hop number h, the retry times r, the average transmission latency of the quickest-reply node t, and the average transmission latency of 1-hop neighboring nodes is T.

The all agreement approach [8] featured a distributed, dynamic host configuration protocol for address assignment called MANETconf, where each node maintains both *Allocated* and *Allocated_Pending* data structures. An *Allocated* address pool records all IP addresses in use. The moment a node joins or leaves a network, the *Allocated* address pool in every node would be refreshed, adding new addresses to or removing leaving nodes' addresses from the list. However, the more nodes in a network are, the larger an *Allocated* address pool becomes. It inefficiently takes enormous storage space and huge efforts to maintain every address pool in each node. The latency of assigning a usable address to a joining node is at most $O(r \times D \times t)$, which performs the worst among all solutions. In MANETconf, a joining node waits for all nodes to respond via affirmative replies for a selected address. If at least one response is negative, another address is retried. Communication overhead is at most $O(r \times n^2)$ because every node uses broadcast messages to maintain both *Allocated* and *Allocated_Pending* data structures for node joining, leaving and networks merging. The greatest communication overhead will be produced during network merging because nodes in networks must perform DAD.

Table 1 The comparison of characteristics and performances

Categories	All agreement	Leader-based	Best-effort	Buddy system	Our solutions	
Solutions	MANETconf [8]	DACP [13]	Prophet [19]	AAAC [14]	DRAA	CRAA
Address allocation	Decentralized	Decentralized	Decentralized	Decentralized	Decentralized	
Network merging	Decentralized	Centralized	Decentralized	Decentralized	Decentralized	Centralized
DAD during configuration	Yes	Yes	No	No	No	
Changing addresses during merges	Duplicate nodes	Duplicate nodes	All nodes	Duplicate nodes	Duplicate nodes	
Evenness	Yes	Yes	Yes	Probably no	Probably no	Yes
Latency of address allocation	$O(r \times D \times t)$	$O(r \times h \times n)$	$O(t)$	$O(t)$	$O(t)$	$O(T)$
Communication overhead	$O(r \times n^2)$	$O(r \times h \times n)$	$O(d \times n)$	$O(r \times n^2)$	$O(r \times n^2)$	$O(r \times h \times n)$
Scalability	Small	Medium	Medium/High	Medium	High	

Among leader-based approaches, Sun and Belding-Royer [13] proposed a Dynamic Address Configuration Protocol (DACP). Each node independently obtains a candidate address and registers the chosen address to the leader node for DAD. Communication overhead is $O(r \times h \times n)$, the number of which is smaller because only the leader node broadcasts the network-merging information. The leader node periodically broadcasts Network Identifier Advertisement messages to detect network partitioning and merging. The biggest drawback Leader-based approaches bear is that the workload of the leader node is too heavy due to DAD for all joining nodes.

The prophet address allocation protocol [19] makes use of an integer sequence consisting of random numbers through the stateful function $f(n)$ for conflict-free allocation. The function $f(n)$ keeps low probability of address duplication. Prophet does not perform DAD to reduce communication overhead during network merging, but nodes with the smaller network identifier (NID) change their IP addresses no matter duplication occurs or not. It is because during network merging, the various addresses of nodes in different networks will cause the function $f(n)$ not to guarantee conflict-free allocation. Although Prophet brings the benefit of lower communication overhead [$O(d \times n)$], nodes break all on-going connections with the smaller NID. When two large networks merge, the impact of connection loss is significant.

Tayal and Patnaik [14] proposed an address assignment for the automatic configuration (named AAAC), to which the buddy system is applied whenever resources run out and new nodes seek to join a network. A node requested to allocate an address broadcasts *SEARCH_ADDR* messages, waiting for all other nodes to respond with their own sets of IP addresses. This node subsequently compiles statistics from responding sets to acquire unused addresses, one of which will be allotted to a newly-joined node, and send back *NODE_CRASH* messages to these nodes, allowing each node to take over neighboring addresses. In order to get an address at $O(t)$, AAAC results in the unbalance of address pools. Free address blocks deplete quickly because a node splits its 2^m-unit block into two with 2^{m-1} units. For example, if a node holds 2^{10} free address block, the block will be consumed after ten times of split-up. After the consumption, the node starts free address searching which leads to longer addressing latency. In AAAC, every node in merging networks broadcasts its IP address and address pool to whole networks for network merging, whose communication overhead is $O(r \times n^2)$.

The best case of addressing latency is $O(t)$ in Prophet, AAAC and our decentralized ring-based address autoconfiguration (DRAA) protocols. All of them allocate a usable address from the free address block immediately when receiving an address request. Our DRAA protocol is a low-latency solution because each node independently

allocates partial IP addresses and does not need to perform the DAD during the node-join operation. The latency is $O(T)$ in our centralized ring-based address autoconfiguration (CRAA) protocol because a joining node waits for all 1-hop neighbors' responses and chooses the largest free address block to use. Contrasted with Prophet and AAAC, the CRAA protocol increases a little latency but postpones the time for resource consumption and leads to the evenness of address allocation. The primary difference among the all agreement approach, the buddy system approach and RAA protocols is that the former two approaches are unlikely to recognize which nodes should take over unused address sets so that blind broadcast messages are transmitted to inform all nodes in the network, whereas RAA protocols resolve the blind broadcast problem by the logical ring among nodes. As a result, our solutions reduce the number of broadcast messages and avoids defects shown in the other two approaches. The main difference between the leader-based approach and the CRAA protocol is that the leader in the former approach presides at DAD, whereas the holder in CRAA is not in charge of DAD.

The remaining sections of the paper are organized as follows. Section 2 proposes preliminary of this paper, which includes model, basic idea and improvement contribution of RAA protocols. In Sect. 3, we present the details of the DRAA with regard to address resource maintenance and node behavior handling. In Sect. 4, the CRAA is introduced. Section 5 shows the performance analysis. Finally, Sect. 6 draws conclusions for the paper.

2 Preliminary

In this section, we proffer both conceptual discrepancies among various protocols and the basic idea of RAA protocols. We present the basic idea of RAA protocols in Fig. 1, and show how to allocate IP address in RAA with the ring structure and the binary buddy system in Fig. 2. The key to determining the effectiveness of a dynamic IP address assignment protocol mainly lies in both the latency of node joining and communication overhead of network merging, and we illustrate the comparison among various protocols in terms of node joining and network merging in Figs. 3 and 4, respectively. We present the difference of address allocation between DRAA and CRAA protocols during node joining in Fig. 5, and show the improvement of resource management in Fig. 6. Finally, in Fig. 7, we illustrate how DRAA and CRAA protocols do with existing connections when duplicate addressing occurs during network merging.

2.1 Model

We consider mobile wireless networks where all nodes use IP address to communicate with others. Such a network can be modeled as follows. A mobile wireless network is represented as a graph $G = (V, E)$, where V is the set of nodes and E is the edge set which gives available communications. The communication (u, v) belonging to E means that u sends messages to $v$connection set of vertex u is defined as $E(u)$. In a given graph $G = (V, E)$, we denote by $n = |V|$ the number of nodes in the network. The neighbor set $N(u)$ of vertex u is defined as $N(u) = \{v|(u, v) \in E\}$. The identifier (ID) and the related list of node u are represented as N_u and RL_u, respectively. The network identifier is represented as NID, which is 2-tuple: <IP address, Random number>, by whose uniqueness is distinguished in different networks. The ID of a node is 2-tuple: <NID, IP address>. The successor, the predecessor and the second predecessor of node u are represented as S_u, P_u, and SP_u.

2.2 Basic Idea

In this paper, we draw on a novel technique developed in peer-to-peer (P2P) networks to provide a logical view for resource maintenance. The buddy system is adopted due to such benefits as quicker responses as well as fewer broadcast messages for both address assignment and resource maintenance, enabling each node to hold its disjoint set of IP addresses. Recently, P2P technology is fast gaining ground around the world. It offers a logical network, allowing clients to share files in a P2P way. One of the distributed hash table (DHT) based approaches in P2P, known as Chord [12], inspires us to utilize its logical view to perform address resource management. In the prototype of Chord, in order to keep load balancing, each node uses the hash

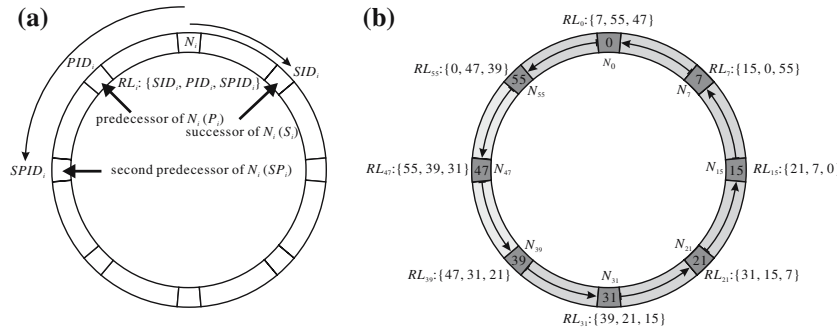

Fig. 1 (**a**) The concept of successor, the predecessor and the second predecessor. (**b**) Each node in RAA protocols maintains its related list (*RL*)

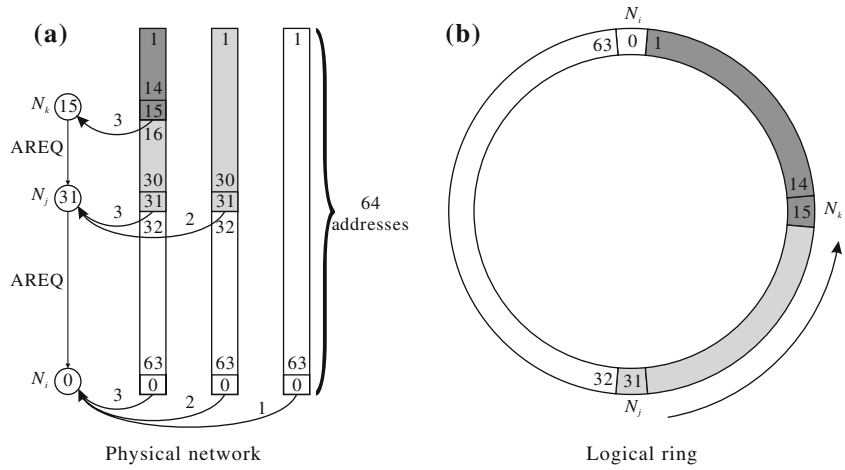

Fig. 2 Using the ring structure and the binary buddy system for IP address allocation

function to produce a node identifier by hashing its own IP address, and each file (i.e., resource) utilizes the same function to produce a key identifier (i.e. both the original key and its image) by hashing its own file name. Unlike Chord, our solutions—RAA protocols—are not necessary to use any hash function to distribute address resources because IP address resources differ from file ones. One file can hold in many nodes whereas an IP address only exists in one node. If the hash function is applied to distribute address resources for the evenness in networks, the complexity of address management will be raised and not be suitable in MANETs. For this reason, the buddy system is combined in RAA protocols to manage resources.

In RAA protocols, each node records its logical neighbors' IDs on the related list (RL). Logical neighbors are the successor, the predecessor and the second predecessor. Notice that the RL is only updated during node joining. As presented in Fig. 1a, if N_i exists in RAA protocols, the successor (S_i) is the node which allocates an address block to it and is also the first node in the clockwise course starting from N_i. The predecessor (P_i) and the second predecessor (SP_i) are the first node and the second node in the anticlockwise course starting from N_i, respectively. The successor's, the predecessor's and the second predecessor's IDs can be represented as SID_i, PID_i and $SPID_i$. The RL of N_i is RL_i: {SID_i, PID_i, $SPID_i$}. As shown in Fig. 1b, each node possesses its own anticlockwise-course free address block. During node leaving, only the successor in the clockwise course has to be informed; then it will retrieve the address block of the leaving node.

Figure 2 shows two illustrations of how to assign IP addresses with RAA protocols. To simplify statements, we use fewer address blocks, decimalize IP addresses and assume there are 64 IP addresses (0–63). In Fig. 2(a), at first node N_i randomly selects an IP address which is presumed as 0 and possesses the whole address block (including 64 IP addresses). Figure 2b is a ring formed through the logical relationship among nodes. When N_i receives an address request (AREQ) from N_j, N_i uses the binary buddy system to split its address

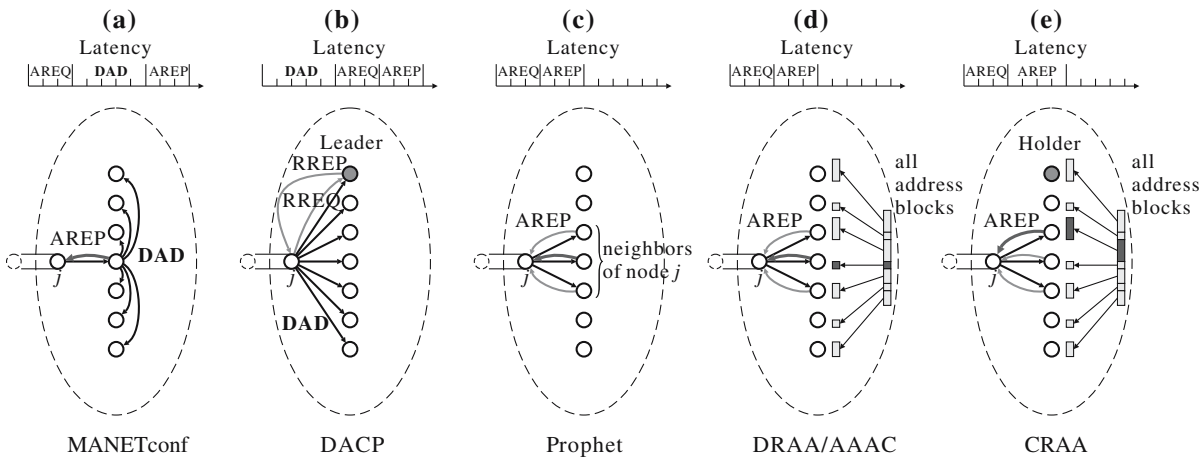

Fig. 3 The comparison of various protocols during node joining

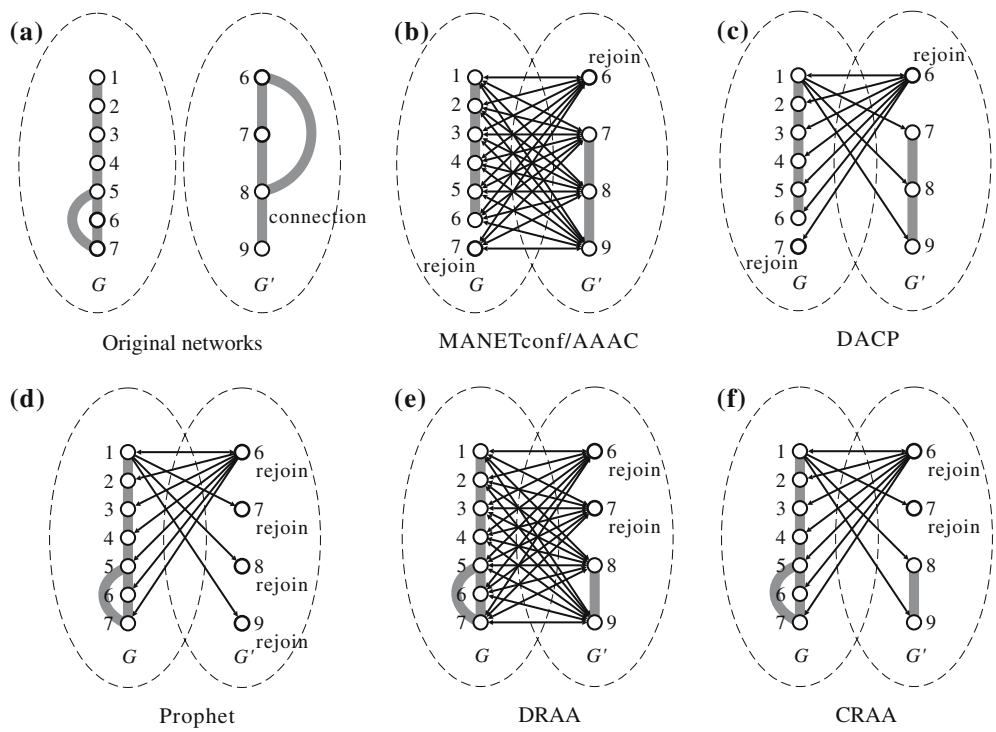

Fig. 4 The comparison of various protocols during network merging

block into two with 32 IP addresses, respectively. From the perspective of the assigning node, N_i assigns the second free address block from the anticlockwise direction to N_j and keeps the other itself. N_j thereupon uses the last IP address of the assigned block. On analogy, after N_j receives an AREQ from N_k, N_j repeats the identical procedure—dividing an address block into two and allotting one to N_k. Procedure finished, N_i, N_j, and N_k all own an IP address and a disjoined address block. All blocks make up a logical ring whose block order indicates the usage situation of address resources and guarantees that every address allocated by nodes does not overlap.

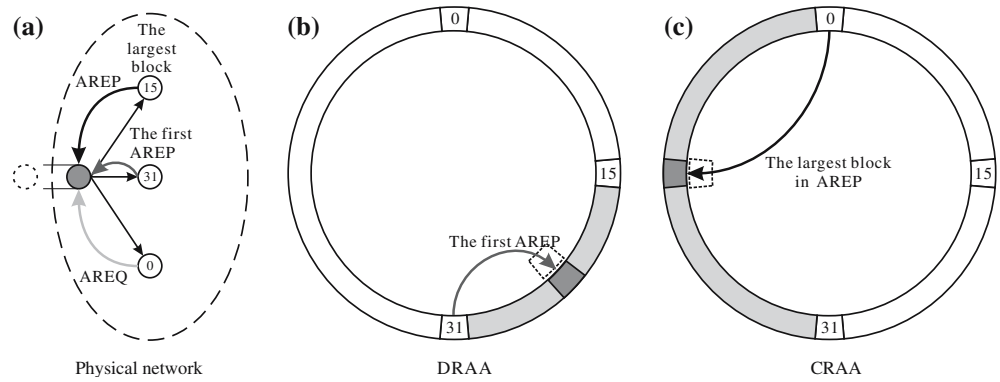

Fig. 5 Address allocation in logical ring of DRAA and CRAA

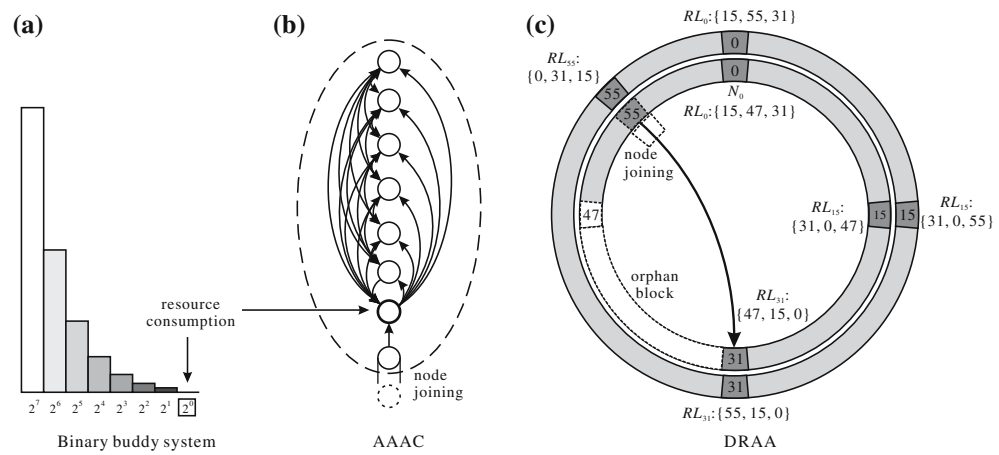

Fig. 6 (**a**) Resource consumption in the binary buddy system. (**b**) Retrieving orphan blocks in AAAC during resource consumption. (**c**) Retrieving orphan blocks in DRAA during node joining

2.3 Improvement contribution

Up to the present, dynamic IP address assignment in MANETs has lacked a perfect solution. In order to obtain IP addresses and maintain usable IP address resources, broadcast methods are largely applied for sending control messages to MANETs. Nevertheless, too many control messages causing heavy traffic load in MANETs solely result in both the reduction of bandwidth utility and the inscalability of current schemes. To offer effective IP address assignment in a dynamic network environment, the solution provided by our addressing protocol presents three goals, aiming at efficient, rapid IP address distribution as well as applicable address resource maintenance: (1) low latency, (2) low communication overhead, (3) evenness, and (4) uninterruptible connection. Namely, low latency produces the results that a requested node timely gets a unique address in the IP address assignment process, communication overhead is lessened to enhance network efficiency, and address resources are evenly distributed in each node. If duplicate addressing occurs during network merging, the on-going TCP connections in duplicate nodes will be broken, which results in poor uninterruptible connection.

2.3.1 Latency

In Fig. 3a, b, MANETconf and DACP are not conflicting free protocols, so they have to perform DAD during node joining, which increases the latency of node joining. On the contrary, the other

protocols, such as Prophet, AAAC, DRAA, and CRAA, are conflicting ones so that they can assign a unique address to new nodes without DAD (Fig. 3c–e). Furthermore, AAAC, DRAA, and CRAA are categorized into buddy system approaches, whose main difference is the evenness of address blocks in all nodes during node joining. The more uneven the size of every address block, the more probable the fact that a new node is not assigned with an address and greater latency occurs during node joining as well. In AAAC and DRAA, a new node N_j broadcasts a one-hop AREQ to its neighbors and awaits the very first address reply (AREP). As shown in Fig. 3e, although selecting the fastest AREP lessens the latency, address blocks are distributed unevenly in the meantime. However, the CRAA protocol awaits AREPs of all neighbors and picks up the largest address block for use, which effectively improves the uneven distribution of address blocks shown in the former case.

2.3.2 Communication overhead

In a dynamic IP address assignment protocol, the largest communication overhead occurs in network merging, during which all nodes perform DAD to confirm that no address is conflicting among networks. When duplicate addresses exist, duplicate nodes need to discard connections and rejoin the network. Figure 4 illustrates the comparison among various protocols in terms of network merging. Figure 4a shows original networks. There are seven nodes and seven connections in the left-sided network (G) whereas four nodes and four connections are indicated in the right-sided network (G'). As shown in Fig. 4b, c, each node in MANETconf and AAAC has to broadcast its own address to the network for DAD, whereas only the leader node broadcasts addresses in DACP. In MANETconf, AAAC and DACP, among duplicate nodes (node 6 and 7), the node with fewer connections is selected to rejoin the network. But duplicate nodes have the same connections (two connections) in two networks, one of them should be made to acquire a different IP address. As shown in Fig. 4d, Prophet does not perform DAD during network merging, in that the function $f(n)$ can not offer new nodes unique addresses if nodes are not programmed to rejoin a network when Prophet executes network merging. Communication overhead is reduced if DAD is not executed; however, all on-going connections are broken in the smaller NID (i.e., assume the G' network has the smaller NID), where all nodes need to rejoin the network. As shown in Fig. 4e, f, only duplicate nodes in the smaller network G' need to rejoin the network in both DRAA and CRAA protocols. The advantage of such is that in the smaller network, the speed to resume connections is higher due to fewer nodes. The DRAA protocol provides a fully distributed solution—each node is required to broadcast its address to the network for DAD. In contrast, in the CRAA protocol, the holder records addresses of all nodes to form a node list; therefore, it merely takes the holder to broadcast the node list to the network for DAD.

2.3.3 Evenness

Discrepancies among various solutions during node joining have shown in Fig. 3. In this part we present how dissimilarly the logical ring in both DRAA and CRAA protocols assigns address blocks during node joining. Figure 5 shows the difference of address allocation between these two protocols. The DRAA protocol uses a free address block immediately upon receiving an AREP, whereas the CRAA protocol waits for all one-hop neighbors' AREPs and chooses the largest free address block for use. As indicated in Fig. 5b, by using DRAA, a new node gets an available address block in no time when joining a network; however, a probable consequence might be the uneven distribution of address blocks. As denoted in Fig. 5c, through employing CRAA, the latency increases because the largest address block is picked; nevertheless, this protocol enables all address resources to disperse more evenly in each node, hence postponing the depletion time of address resources.

The binary buddy system is one common buddy systems with a resource size of 2^m units and starting off this size. When the application issues a request, a 2^m-unit block is split into two with 2^{m-1} units, respectively. However, resources in the binary buddy system are depleted rapidly. As displayed in Fig. 6a, when a node owns 2^7 addresses and allocates them to other nodes up to seven times,

address resources will be consumed. As shown in Fig. 6b, as a new node asks a resource-consumed node for an address, the latter broadcasts messages to all nodes in order to retrieve unused blocks (orphan blocks) and searches for a free address block for the new node. In the traditional buddy system approach (AAAC), orphan blocks are retrieved solely during resource consumption. If a network changes with high frequency, then resources usually run out. If so, the latency of a new node's requiring an available address block inevitably increases. The failed nodes are checked for their failure and orphan blocks are recovered during the node joining stage in our approach. Without the extra actions of failure-nodes collection, the resource can be easily reused on the node joining stage. The resource consumption is decreased, the latency of address requirement is decreased, and orphan blocks are retrieved. For example, if the allocated node 47 has been failed and marked white as illustrated in Fig. 6c, then a new node 55 of the inside ring joins the network and obtains RL_0 from N_0. Via PID_0 (47) and $SPID_0$ (31) on the RL_0, orphan blocks are retrieved and become parts of the logical ring, shown as the outside ring. During node leaving, it postpones the resource consumption time and avoids over-abundant broadcasting messages to use the RL to retrieve orphan blocks. Our solution achieves highly efficient resource management and address allocation through the combination of logical ring and the buddy system.

2.3.4 Uninterruptible connection

Comparisons among different solutions during network merging have shown in Fig. 4. Here is explicitly stated how DRAA and CRAA protocols do with existing connections upon encountering duplicate addresses during network merging. As illustrated in Fig. 7a, during network merging, all nodes in both networks are required to broadcast their IDs as well as connection numbers in the DRAA protocol. Each node is informed of full information of both networks: that is, the ID and the connection number of each node, and the respective node number in these two networks. As shown in Fig. 7b, in the CRAA protocol, since the holder owns all node IDs and node numbers, solely the holder needs to broadcast network information. Whether DRAA or CRAA is applied, all nodes obtain full information in network merging. As indicated in Fig. 7c, when nodes get needed information, N_{39} in network G and network G' finds out there is a duplicate address in the network that is going to merge. N_{39} in network G' identifies the connection number is equal to that in network G (there is one connection in N_{39}) and also the node number is fewer in network G'; therefore, it rejoins the network to change its IP address. Presume that N_{39} asks N_0's permission to reenter, the address of N_{39} will be altered as 52 and reconnect with N_{21}. Because we select the duplicate node in the smaller network to rejoin, it is more likely to fast re-connect.

3 Decentralized Ring-Based Address Autoconfiguration Protocol (DRAA)

Since nodes in MANETs have joining, leaving, partitioning and merging behaviors, this section sheds light on how the DRAA protocol handles these behaviors. Before we introduce it, the state diagram of RAA protocols should be shown first to help handle the node behaviors. The state diagram of RAA protocols distinguishes these behaviors and is detailed in Fig. 8. There are totally five states in RAA protocols: INITIAL, STABLE, MERGE, HS, and FINISH. When a new node intends to join a network, it enters the INITIAL state and waits for a free address block. A node enters the STABLE state when getting an address. If nodes are informed of network merging by some node, they enter the MERGE state for merging. If the holder leaves the network, other nodes enter the holder selection (HS) state to select a new holder. If leaving the network, a node enters the FINISH state. According to node behaviors, we realize how the DRAA protocol deal with joining and leaving. In following paragraphs, we will state the initiation of networks and specify node behaviors via the DRAA protocol.

3.1 Initial state

A node is in the INITIAL state before getting a usable address. At the start, the first node, N_i, enters a network and broadcasts a one-hop AREQ

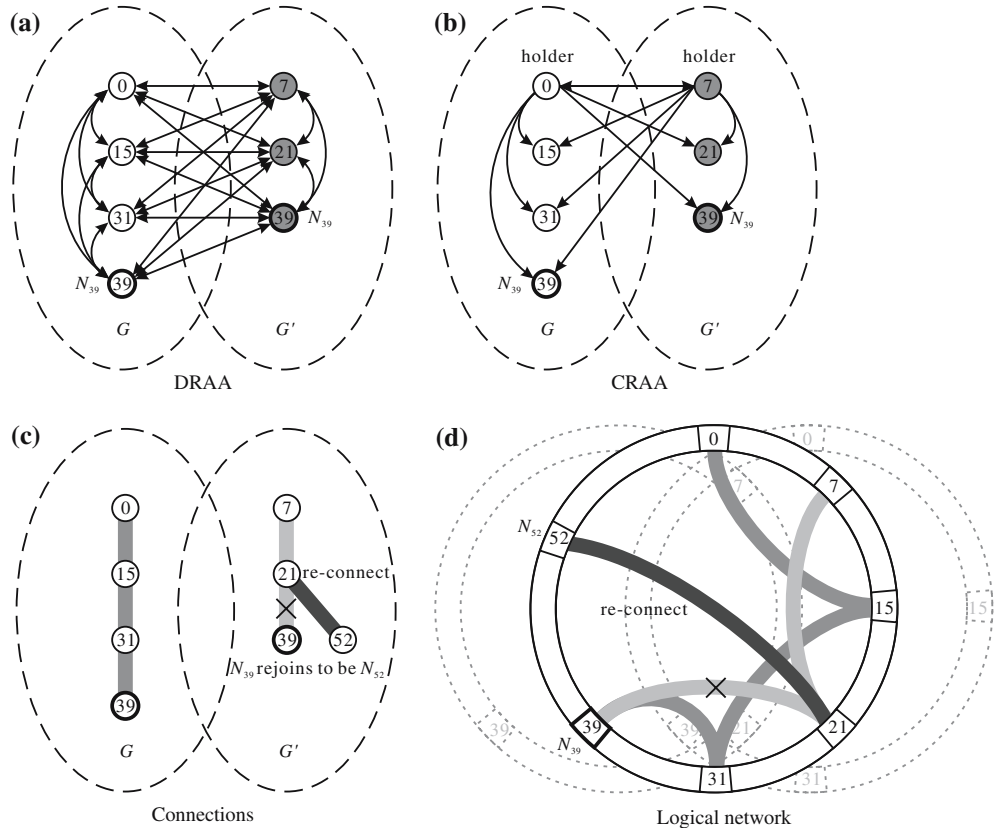

Fig. 7 Connection variation during network merging

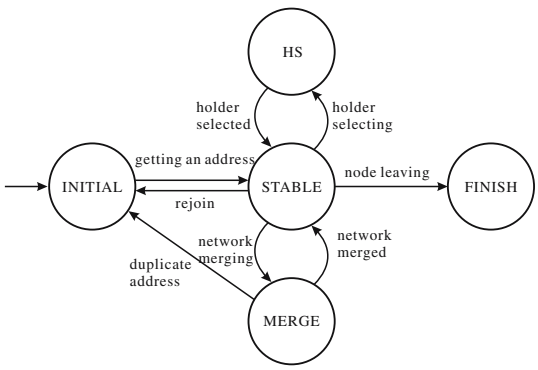

Fig. 8 The state diagram of RAA protocols

message. N_i awaits an address request timer (AREQ_Timer) to gather responses from other nodes. Once the timer expires and N_i does not get any response, N_i gets its identity as the first node (i.e., the holder) in the network. N_i randomly chooses an IP address, and sets it into its ID. The holder uses its ID and a random number as the NID. The NID is periodically broadcast to the whole network by the holder, enabling nodes to get the NID of their locating network and to detect network partitioning and merging. When the NID holder fails, there is at least a node in the designated network group not reviving the NID after one broadcasting period. In this scenario, the network partition processing will be started.

3.2 Node joining

When aiming to enter a network, N_j needs to secure an IP address and then checks whether P_j exists in the network in order to ensure the wholeness of address blocks. The joining procedure consists of two phases: (1) address requesting and (2) failed node checking. The address requesting phase is used to allocate an IP address to a new node

whereas the failed node checking is used to check whether the predecessor of the new node is alive.

3.2.1 Address requesting phase

With N_j in the initial state and intending to join the network in DRAA, the address requesting phase will be triggered. The address requesting phase is applied to allocate an IP address and an address block to a new node, including an AREQ issued from the new node, an AREP replied from neighbors of the new node and an address reply acknowledgment (AREP_ACK) sent by the new node to the neighboring node which is the first to transmit an AREP. The steps of the address requesting phase are given below.

A1: Let N_j be in the initial state and intend to join the network in DRAA. It sends an AREQ message to the neighbor set $N(j)$ and awaits an AREP.

A2: Upon getting the quickest AREP message, N_j immediately takes over the free address block distributed from the responding node N_i, uses the last address from the address block as its own IP address, and modifies RL_i to be its own RL_j. The original RL_i: {SID of N_i , PID of N_i, SPID of N_i} can be modified as RL_j: {SID_j = ID of N_i, PID_j = PID of N_i, SPID_j = SPID of N_i}.

A3: After N_j takes over the address block, N_j transmits an AREP_ACK to the responding node. An AREP_ACK message is used for ensuring that N_j receives an AREP message and participates in the network.

A4: If N_j does not get any AREP message after an AREQ_Timer expires, N_j retries the address requesting phase r times. r is the maximum of retry times.

A5: If N_j retries the address requesting phase r times and still not receiving any AREP message, N_j randomly selects an address to use and possesses all address blocks.

As shown in Fig. 9a, upon entering a network, a node broadcasts a one-hop AREQ message and waits for AREP messages from its neighbors. An AREP message consists of what a responding node distributes: (a) NID of the responded node, (b) RL of the responded node and (c) the free address block. According to the AREP from N_i, N_j knows not merely its own logical position in RAA protocol but also the fact that both S_j and P_j are N_i and SP_j is N_j itself. For example, in Fig. 9b, N_{21} modifies RL_{31} : {0, 15, 0} to be its own RL_{21}: {31, 15, 0}.

3.2.2 Failed node checking phase

After the address requesting phase, N_j uses the IP address to communicate with other nodes in the MANET and enters the STABLE state from the INITIAL state. The Failed node checking phase is used to check whether the predecessor of the new node is alive. The new node first sends an alive unicast checking (ACHK) message to its predecessor. If the predecessor is alive, the predecessor replies with an ALIVE message to the new node. If the predecessor fails, the new node sends an unicast address retrieve (ARET) message to its second predecessor to retrieve the address block of the predecessor. Then the second predecessor sends back an address retrieve acknowledgment (ARET_ACK) to the new node to inform it which ones are its new predecessor and second predecessor. The design goal of our DRAA is only to tolerate a single-node failure since each node only keeps the information of successor, predecessor and the 2nd predecessor. The address is requested by the unicast communication to the predecessor and the 2nd predecessor. Therefore, our strategy obtains the lower communication time but causes the single-fault tolerant result. We may easily modify the DRAA protocol to increase the fault tolerance ability by keeping more backup links. However, this result is not our main concern, since the keeping the information of the more backup links information incurs the more extra overhead. The operations of the failed Node Checking Phase are given below.

B1: N_j sends an ACHK message to P_j (PID_j is in its RL_j).

B2: If P_j is alive, it sends an ALIVE message to N_j. The failed node checking phase is finished.

B3: If P_j is invalid, N_j retrieves the address block of P_j. Then N_j sends an ARET message to SP_j.

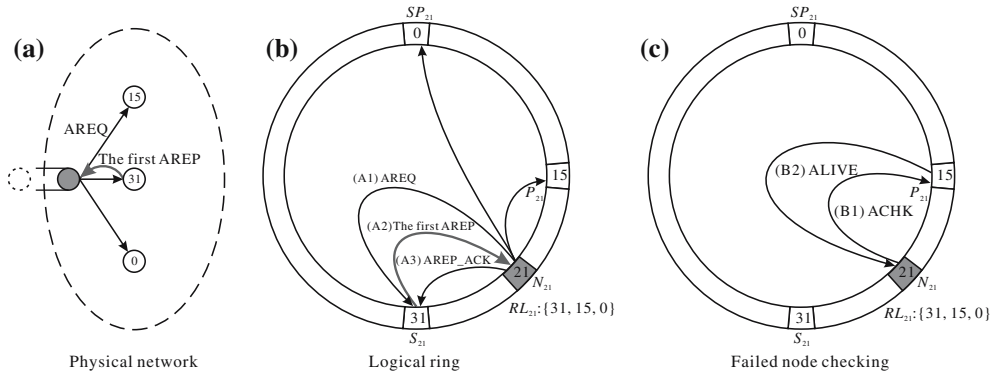

Fig. 9 The message flows of the joining procedure without the node's failure in the DRAA protocol

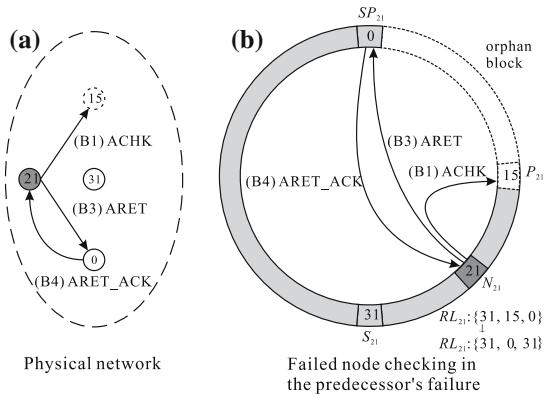

Fig. 10 The message flows of failed node checking phase in the predecessor's failure in the DRAA protocol

B4: An ARET_ACK message is replied when SP_j is alive. An ARET_ACK message includes the RL of SP_j, so that N_j can modify its PID_j and $SPID_j$ in its RL_j. The failed node checking phase is finished.

As seen in Fig. 10, N_{21} (ID = 21) sends an alive checking (ACHK) message to P_{21} ($PID_{21} = 15$). P_{21} fails and its orphan block is retrieved by N_{21}, which will send an ARET message to SP_{21} to notify SP_{21} of P_{21}' failure. Then SP_{21} transmits an ARET_ACK to N_{21}, which will retrieve the orphan block and modify its $RL_{21} : \{31, 15, 0\}$ to $RL_{21} : \{31, 0, 31\}$. Each node in the DRAA protocol has the capability to retrieve its first one anticlockwise orphan block.

3.3 Node leaving

When a node leaves the network genteelly, the leaving node sends a LEAVE message to its successor, and then the successor takes over the address block of the leaving node. Notice that during node leaving, the successor is unnecessary to be the one-hop neighbor of the leaving node, which has to procure the successor's routing path in the routing table. If nodes crash without a LEAVE message, the orphan block of the crashed node is retrieved in the failed node checking phase.

C1: The leaving node N_l sends a LEAVE message to S_l (SID_l is in its RL_l). The LEAVE message includes its ID and RL_l.

C2: When getting the LEAVE message, S_l takes over the leaving node's address block and modifies its own RL.

As shown in Fig. 11, the leaving node N_{47} sends a LEAVE message to its successor S_{47} (N_0), and then N_0 takes over the address block of N_{47}. If nodes crash, due to the ordering of nodes in the DRAA protocol, one node's failure can be restored. As mentioned earlier, when processing the failed node checking phase, the new node retrieves the orphan block of its predecessor.

3.4 Network partitioning

The only responsibility of the holder node in the DRAA protocol is to broadcast its NID. If there is a node not receiving the NID after one broadcasting

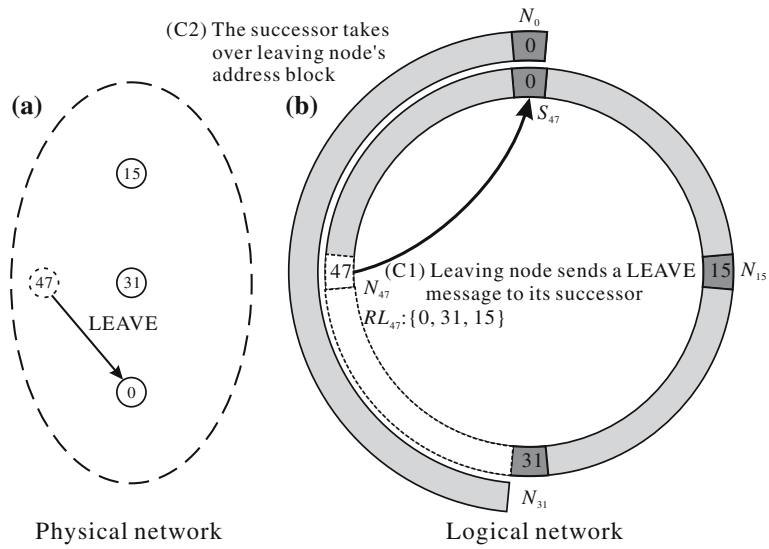

Fig. 11 A leaving node notifies its successor to take over its address block

period, the node may have been partitioned from the network or the NID holder may have been failed. But wireless communication is not reliable; the NID may not be received due to packet loss. To reduce the impact, we define network partitioning as three broadcasting periods without receiving the NID. Once a node detects network partitioning, it performs the holder selection algorithm.

In the holder selection algorithm, we borrow the random backoff in carrier sense multiple access with collision avoidance (CSMA/CA) protocol of IEEE 802.11 standard and slightly modify the idea. When network partitioning is detected by the node, the detecting node enters the HS state and chooses a random backoff timer (RB_Timer) which determines the amount of time the node must wait until it is allowed to broadcast its NID. The RB_Timer reduces the collision of NID messages, broadcast by nodes in the holder selection state.

D1: When N_j detects network partitioning, it enters the HS state and chooses a RB_Timer which determines the amount of time the node must wait until it is allowed to broadcast its NID.

D2: If N_j does not receive any NID, it broadcasts its own NID and waits for other nodes to respond their IDs after an random backoff timer expires.

D3: If N_j receives another NID in its random backoff period, it will broadcast its ID and not send its own NID out. If more than one NID is received, N_j chooses the bigger random number which broadcasts the NID to be its holder.

D4: When a holder is selected, all nodes enter the STABLE state.

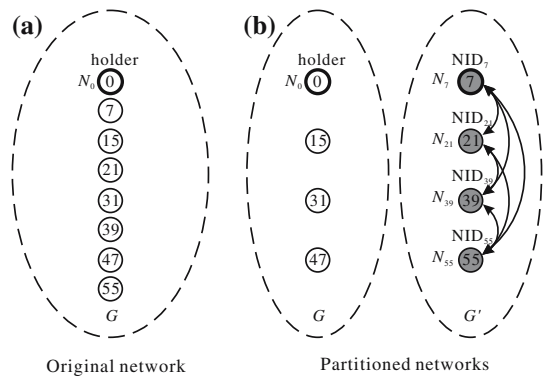

Fig. 12 The holder selection in RAA protocols

Figure 12 shows the original network G is partitioned to two networks G and G'. G still has its holder N_0, so that nodes in G do not need to select a new holder. Nodes in the partitioned network G' can not receive the NID from N_0. Nodes in G' enter the HS state and intend to select a new holder. We assume that the NID_7 of N_7 has the bigger random number than other NIDs. N_7 is selected to be the new holder in G' and the NID of G' is NID_7 which is broadcasted from N_7.

3.5 Network merging

Assume that there are two networks G (NID_G) and G' ($\mathrm{NID}_{G'}$) intend to merge. Network G has n nodes and network G' has n' nodes. In the DRAA protocol, all nodes during network merging have to broadcast its ID for DAD. When all nodes get full information of networks, they choose a holder who broadcasts a bigger NID. If duplicate addressing occurs, duplicate nodes which have fewer TCP connections or are in the smaller network should rejoin the network. The benefits of choosing the node from a smaller network are quicker responses and lesser possible disconnections.

E1: N_j is in network G and receives $\mathrm{NID}_{G'}$, which is different from its own NID_G, and then the network merging is detected.

E2: When merging is detected, the detecting node N_j broadcasts a merging request (MREQ) to all networks.

E3: V and V' (all nodes in both networks) receive the MREQ and then enter the MERGE state.

E4: V and V' broadcast their IDs and the number of TCP connections in both networks; thus, V and V' receive full information of both networks and modify their RLs and choose the holder who broadcasts the bigger NID to be their holder.

E5: If duplicate addressing occurs, duplicate nodes N_d and $N_{d'}$ check the number of TCP connections and node numbers in both network.

- **F1:** If $E(d) > E(d')$ (the number of TCP connections), $N_{d'}$ enters the INITIAL state to request a unique address.
- **F2:** If $E(d) < E(d')$, N_d enters the INITIAL state to request a unique address.
- **F3:** If $E(d) = E(d')$, but $n \geq n'$, $N_{d'}$ enters the INITIAL state to request a unique address.
- **F4:** If $E(d) = E(d')$, but $n < n'$, $N_{d'}$ enters the INITIAL state to request a unique address.

As shown in Fig. 13, there are two networks G (NID_G) and G' ($\mathrm{NID}_{G'}$) intending to merge and they have the same number of nodes. The holders are N_0 and N_7 in G and G', respectively. During network merging, all nodes in both networks broadcast their IDs and get the full network information. We assume there is no duplicate node occurring to simplify the explanation of network merging. NID_G is bigger than $\mathrm{NID}_{G'}$, so that the holder of merged network G is N_0.

4 Centralized Ring-Based Address Autoconfiguration Protocol(CRAA)

With regard to evenness, the centralized RAA protocol (named CRAA) is introduced to distribute address resources evenly. The difference between DRAA and CRAA protocols in the address requesting phase of node joining is that the CRAA protocol selects the largest free address block to use during the address requesting phase. Furthermore, the CRAA protocol in the failed node checking phase can recover more than one failed node because the holder in CRAA maintains the node list (NL) which contains all used IP addresses in the network. The NL is helpful when there are two or more continuous node failures in a ring, so that the lost address resources can be retrieved with the help of the holder. During networks merging, the NL reduces broadcast messages by exchanging the holder's NL only.

In the address requesting phase, the new node in the CRAA protocol waits for all neighboring nodes' AREP messages and selects the largest free address block for use. The DRAA protocol can distribute valid addresses immediately, but the address block of each node will probably not be evenly for the long-term perspective. The advantage of the CRAA protocol is averaging the number of free address blocks managed by each node. Using the CRAA protocol to allot address blocks postpones the address depletion timing in each single node and further manages address resources effectively. By contrast with the DRAA protocol, N_j sends an AREQ message and awaits the expiration of an AREQ_Timer in the CRAA protocol. After the timeout, if N_j gets any AREP message, N_j selects the responding node which distributes the largest free address block. The address requesting phase in the CRAA protocol is given below.

G1: Let N_j be in the initial state and intend to join the network in the CRAA protocol. It

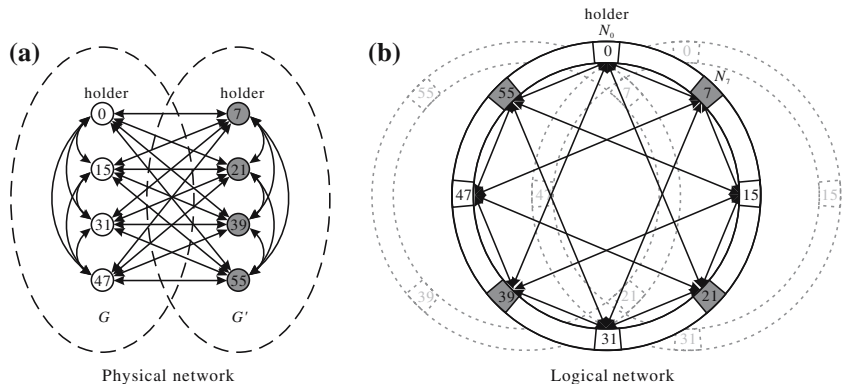

Fig. 13 Network merging in the DRAA protocol

sends an AREQ message to $N(j)$ and awaits an AREP.

G2: Upon getting the largest address block of an AREP message, N_j takes over the free address block distributed from the responding node N_i, using the last address from the address block as its own IP address.

G3: After N_j takes over the address block, N_j transmits an AREP_ACK to the responded node. The AREP_ACK message is used for ensuring that N_j receives the AREP message and participates in the network.

G4: If N_j does not get any AREP message after an AREQ_Timer expires, N_j retries the address requesting phase r times. r is the maximum of retry times.

G5: If N_j retries the address requesting phase r times and still not receiving any AREP message, N_j randomly selects an address to use and possesses all address blocks.

As shown in Fig. 14, a new node intends to join the network. The new node broadcasts an AREQ message to its neighbors (N_0, N_{15}, and N_{31}) and then chooses the AREP from N_0 which has the largest address block to use. After taking over the free address block, the new node becomes N_{47} and sends an AREP_ACK message to N_0.

Since the holder maintains the NL of the network in the CRAA protocol, each node sends an address registration request to the holder when a node takes over a free address block. After a new node uses a new IP address, the new node sends a registration request (RREQ) to the holder that records the new node's address on the NL and the holder sends a registration reply (RREP) to the new node. The CRAA protocol adds two messages, RREQ and RREP, to maintain the NL. The NL can help to recover the failed nodes. If there are two or more continuous node failures in a ring, the lost address blocks can be retrieved with the help of the holder in the CRAA protocol. The failed node checking phase of CRAA protocol is given below.

H1: N_j sends an ACHK message to P_j (PID_j is in its RL_j).

H2: If P_j is alive, it sends an ALIVE message to N_j.

- **I1:** N_j sends an RREQ to the holder to update the NL.
- **I2:** The holder sends back an RREP to N_j. The failed node checking phase is finished.

H3: If P_j is invalid, N_j retrieves the address block of P_j. Then N_j sends an ARET message to SP_j.

- **J1:** An ARET_ACK message is replied when SP_j is alive. The ARET_ACK message includes the RL of SP_j, so that N_j can modify its PID_j and SPID_j in its RL_j.
- **J2:** N_j sends an RREQ to the holder to update the NL.
- **J3:** The holder sends back an RREP to N_j. The failed node checking phase is finished.

H4: If SP_j is invalid too, N_j sends an RREQ to the holder to inform that all predecessors fail.

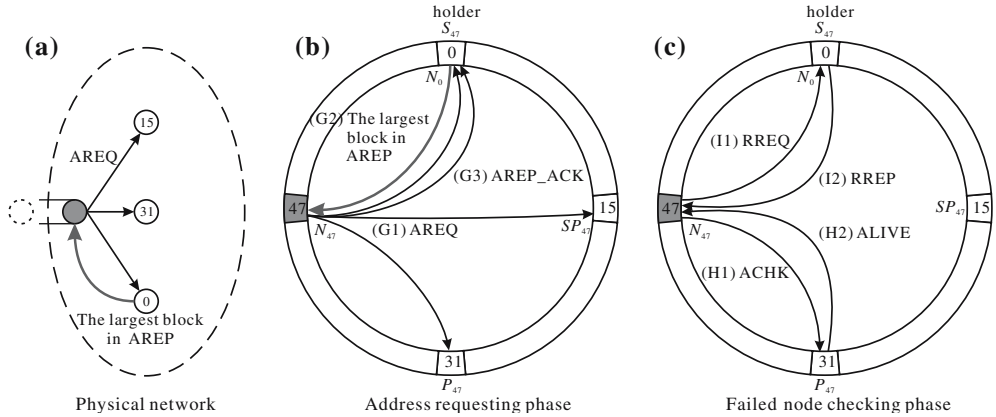

Fig. 14 The message flows of the joining procedure without the node's failure in the CRAA protocol

H5: The holder sends back a RREP to N_j which includes the new predecessors of N_j. The failed node checking phase is finished.

As shown in Fig. 15a, the predecessor P_{47} of the new node N_{47} fails. When a new node retrieves the ARET_ACK message from SP_{47}, it sends an RREQ message to the holder N_0 to remove the invalid address PID_{47} (ID is 31) from the NL in the CRAA protocol. As shown in Fig. 15b, all predecessors of N_{47} fail. N_{47} sends an RREQ message to N_0 to remove the invalid addresses [PID_{47} (ID is 31) and $SPID_{47}$ (ID is 15)] from the NL. The holder N_0 sends back an RREP message to N_{47} for updating all predecessors of N_{47} in its RL_{47}.

By contrast with the DRAA protocol during network merging, in the CRAA protocol, when the holder receives the MREQ, it broadcasts its own NL. Each node receives the holders' NLs, merges those NLs, modifies its own RL and chooses the bigger NID to be its network identifier. As shown in Fig. 16, there are two networks G and G' intending to merge. Only the holders N_0 and N_7 broadcast their own NLs to G and G'. This method reduces large communication overhead during network merging. The NL becomes much significant in CRAA, so that we will introduce the replication of the NL.

The NL incorporates in all used nodes in the network and is a helpful information source for both the lost address block retrieval and network merging. If the NL is lost in holder crashing, it requires the assistance from all nodes to rebuild the NL. In order to lessen the information loss caused by critical node crashing, replication—extracted from P2P—is used to make several replicas of important information to different nodes. Each node records an RL in RAA protocols, and the holder has no exception. The NL replicas exist in the holder's successor, predecessor and second predecessor. When the holder gets an RREQ message, the holder copies the latest NL to these three nodes to ensure the freshness of the NL. If any one of them does not get the NID from the holder twice, it broadcasts a new NID. Each node knows the holder changes when it receives the new NID. There is no need to perform the holder selection algorithm in this situation. Node partitioning is indicated when a node fails to get an NID three times in a row, whereas holder changing occurs if a node with an NL fails to obtain an NID two times successively. Therefore, the holder selection algorithm is not performed in a partitioned network where a node has the replication of the NL, which reduces the probability of holder selection and results in fewer control messages.

5 Performance Analysis

To make appropriate protocol simulation, all protocols are implemented with C++. We select AAAC [14] and Prophet [19] to compare with our protocols, and do not consider comparing with MANETconf and DACP. This is because both of AAAC, Prophet are conflicting free protocols,

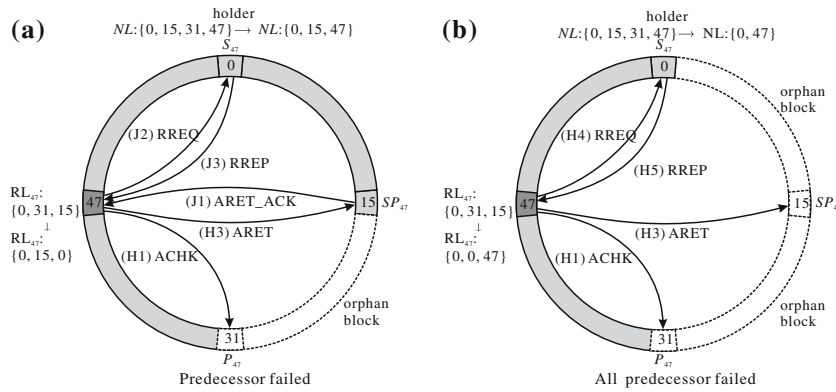

Fig. 15 The message flows of failed node checking phase in the predecessor and second predecessor's failures in the CRAA protocol.

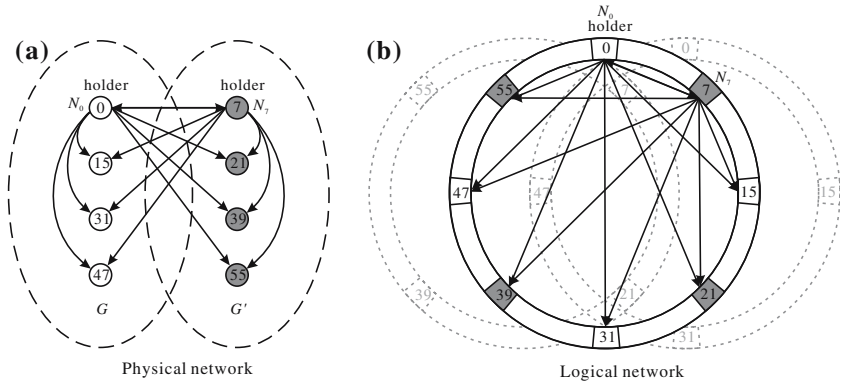

Fig. 16 Network merging in the CRAA protocol

while MANETconf and DACP are not. In AAAC, the authors had shown its performance is better than MANETconf. Both of MANETconf and DACP have long latency when new nodes request addresses.

The simulation parameter is shown in Table 2. The simulation time is 1,500 s. The speed of nodes is varied from 0 to 10 m/s, and the pause time is set to 0 for strict reasons. The mobile nodes move according to the random waypoint mobility model [1]. The network size is set to $1,000 \times 1,000\,\mathrm{m}^2$ where the number of nodes is from 50 to 300. The size of address blocks is set to 2^m IP addresses and m are 10 and 24. To account for our solution in a simple fashion, we apply private IPv4 addresses. RAA protocols also adapt easily to the IPv6 address space. The maximal size of the address block is 24-bit (class A) which contains 16,777,216 IP addresses and makes a suitable selection for large networks. In particular, in our simulation, the transmission range is 250 m. The underlying routing protocol applies the DSDV [9] in all simulations, but our solutions can be used with any routing protocol. The network is initialized with a single node. Nodes join the network every 1 s and arriving nodes are placed randomly in the rectangular region. The beacon interval for the holder to broadcast its NID is set to 10 s. The maximum retry times r of AREQ is set to 3 [19], and the AREQ_Timer is set to 2 s for quicker addressing. The number of TCP connections is set to 20% of nodes. For strict reasons, when all nodes are stable, node leaving and rejoining, network partitioning and network merging are produced freely. The performance metrics of the simulation are given below.

- *Latency*: Latency of address allocation represents the average latency for a new node to obtain a unique IP address within the network. The shorter the latency, the better, since it means a new node can get a usable IP address more rapidly.
- *Communication overhead*: Communication overhead refers to the number of control messages transmitted during the simulation period, including unicast and broadcast messages.

Table 2 Simulation parameters

Parameters	Value
Network size	$1,000 \times 1,000\,m^2$
Number of nodes	50–300
Address block size	2^{10} and 2^{24}
Inter-arrival time interval	1 s
Maximum node speed	10 m/s
Transmission range	250 m

Normally, broadcast messages occupy more bandwidth than unicast messages do. In our simulation, we investigate statistics of the number of both unicast and broadcast messages.

- *Evenness*: Evenness implies that address blocks should be evenly distributed in all nodes, which indicates that each node has the capability to assign address blocks to newly joined nodes. The more evenly address blocks are allotted, the fewer the address resource consumption times in each node are, from which we are able to determine whether address blocks are evenly distributed. Also, latency is lengthened if any resource consumption is produced.
- *Uninterruptible connection*: If duplicate addressing occurs during network merging, the on-going TCP connections in duplicate nodes will be broken. Duplicate nodes thus need to re-transmit data, which inevitably causes the unnecessary consumption of bandwidth.

5.1 Performance of latency

Figure 17 depicts the average latency of address allocation (axis-*Y*) versus the varied number of nodes (axis-*X*), with two address block sizes: 2^{10} and 2^{24}, respectively illustrated in Fig. 17a, b. In general, the DRAA protocol has the shortest latency because the DRAA protocol fast replies an address request and maintains orphan address blocks when a node joins the network. This strategy not only reduces the resource consumption times in a single node but also shortens the latency. The CRAA protocol has longer latency than AAAC and DRAA protocols because it awaits AREPs of neighbors to select a larger address block. Prophet has the longest latency when the number of nodes is more than

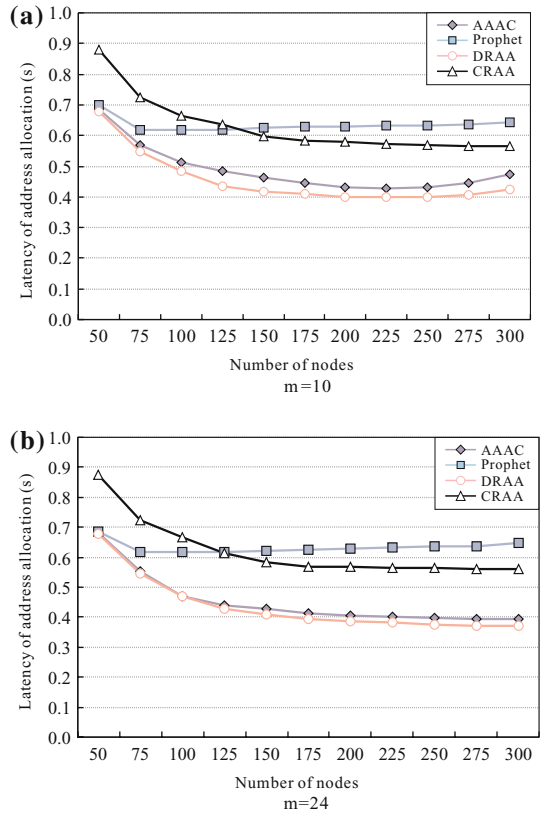

Fig. 17 Average latency of address allocation with the varied number of nodes at $m = 10$ and 24

150 because during network merging, all nodes in the smaller NID have to rejoin the network and wait for the expiration of an AREQ_Timer. If the number of nodes is 50, all protocols have longer latency because the network size is $1,000 \times 1,000\,m^2$ and the transmission range is 250 m. When a new node joins the network, it has the higher probability that no node is within its transmission range, so that the new node needs to await the AREQ_Timer expiration and retry three

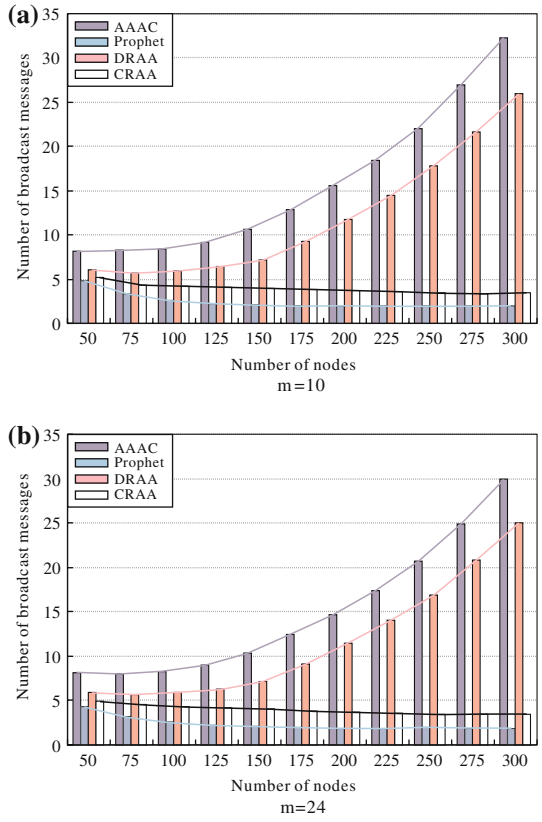

Fig. 18 The number of broadcast messages with the varied number of nodes at $m = 10$ and 24

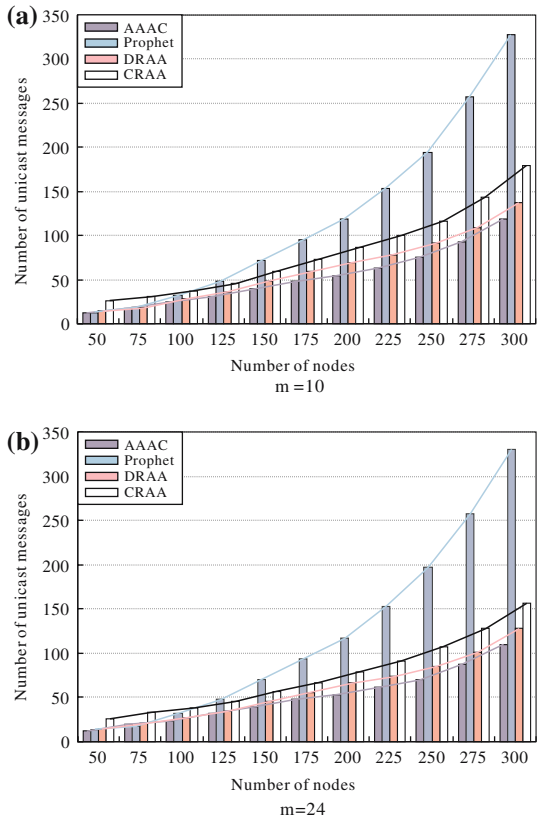

Fig. 19 The number of unicast messages with the varied number of nodes at $m = 10$ and 24

times. When the number of nodes is more than 150 at m = 10 in Fig. 17a, the latency of AAAC and DRAA increases with the node number because more resource consumption times in the whole network will expand the number of address requesting phase retries.

5.2 Performance of communication overhead

Figure 18 shows the number of broadcast messages (axis-Y) versus the varied number of nodes (axis X), with two address block sizes: 2^{10} and 2^{24}, respectively, illustrated in Fig. 18a, b. Most broadcast messages are arisen due to DAD during network merging. Prophet has the fewest broadcast messages because it does not preform DAD during merging and all nodes with the smaller NID rejoin the network. The CRAA protocol has the second fewest broadcast messages because only the holder broadcasts NL for DAD during network merging. The DRAA protocol has fewer broadcast message than AAAC because the resource consumption times of DRAA are less than AAAC. With resources consumed, AAAC needs to recover orphan blocks by means of many broadcast messages. In addition, in node leaving, a leaving node in the DRAA protocol notifies only its successor. However, in AAAC, it needs to notify all nodes of its leaving by broadcast messages because the leaving node does not know which one should take over its address block. On the whole, when the address block size is 2^{24}, DRAA and AAAC protocols have fewer broadcast messages because the resource consumption times are reduced meanwhile.

Figure 19 displays the number of unicast messages (axis-Y) versus the varied number of nodes (axis-X), with two address block sizes: 2^{10} and 2^{24}, respectively, illustrated in Fig. 19a, b. Although the number of unicast messages has fewer impacts

on communication overhead, we investigate it to observe the characteristics of different protocols. AAAC has the fewest unicast messages because most of its control messages are handled with broadcast messages. The DRAA protocol improves control messages in AAAC. For instance, broadcast messages are replaced with unicast messages during node leaving; therefore, its number of broadcast messages is less than AAAC's whereas it has more broadcast messages than AAAC. In the CRAA protocol, more unicast messages are added to maintain the NL, so its unicast messages are more than those in DRAA as well as AAAC. Unicast messages in Prophet are the most because many nodes need to rejoin the network during network merging, which results in the large number of unicast messages. Generally speaking, as the address block size is larger (Fig. 19b), the number of unicast messages is less than that with the smaller address block size. However, the difference of these two numbers is not noticeable. The main discrepancy is that when address blocks are large enough, the resource consumption times in a single and duplicate addressing are reduced, and so is the number of unicast messages.

5.3 Performance of evenness

Figure 20 shows the resource consumption times (axis-Y) versus the varied number of nodes (axis-X), with two address block sizes: 2^{10} and 2^{24}, respectively illustrated in Fig. 20a, b. In AAAC, when a new node requests an IP address from a resource-consumptive node which has no address to allocate, the resource-consumptive node broadcasts *SEARCH_ADDR* messages, waiting for all other nodes to respond their own sets of IP addresses. This shows that if there are more resources running out, the latency of address allocation and broadcast messages will increase. The DRAA protocol, during node joining, has the failed node checking phase to retrieve orphan blocks, so the resource consumption times decrease. Normally, evenly distributed address resources will reduce the resource consumption times. The CRAA protocol has evenly distributed resources because nodes in the CRAA protocol request address blocks from all neighbors and choose the largest one to use. Even though the CRAA protocol raises latency slightly, results show the strategy is worthy. Although the way Prophet allots addresses is different from that of buddy system approaches (AAAC, DRAA, and CRAA) and does not guarantee every allotted address is unique (nevertheless, buddy system approaches do), $f(n)$, which distributes addresses, does not have the phenomenon of resource consumption in Prophet solution. When the address block size is large enough (Fig. 20b), the resource consumption times will be significantly reduced and it is the CRAA protocol whose resource consumption times are close to be 0.

5.4 Performance of uninterruptible connection

Figure 21 depicts the number of breaking on-going connections (axis-it Y) versus the varied number of nodes (axis-X), with two address block

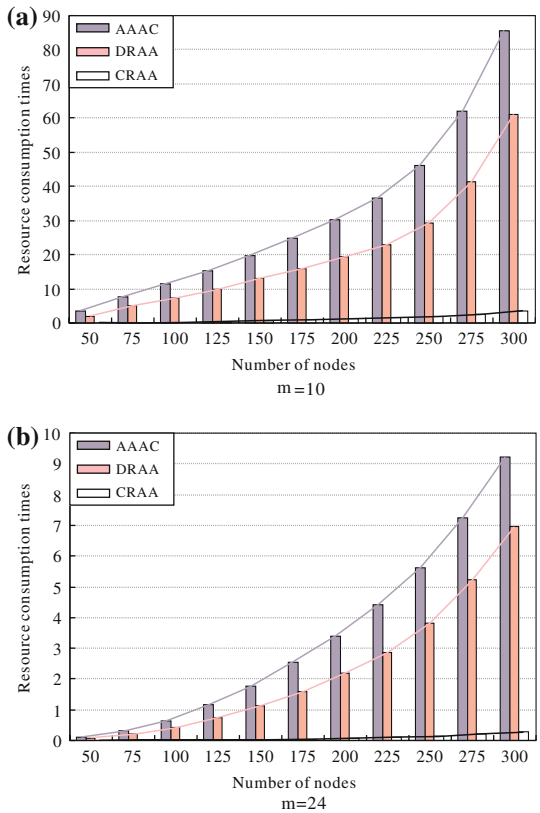

Fig. 20 Resource consumption times with the varied number of nodes at $m = 10$ and 24

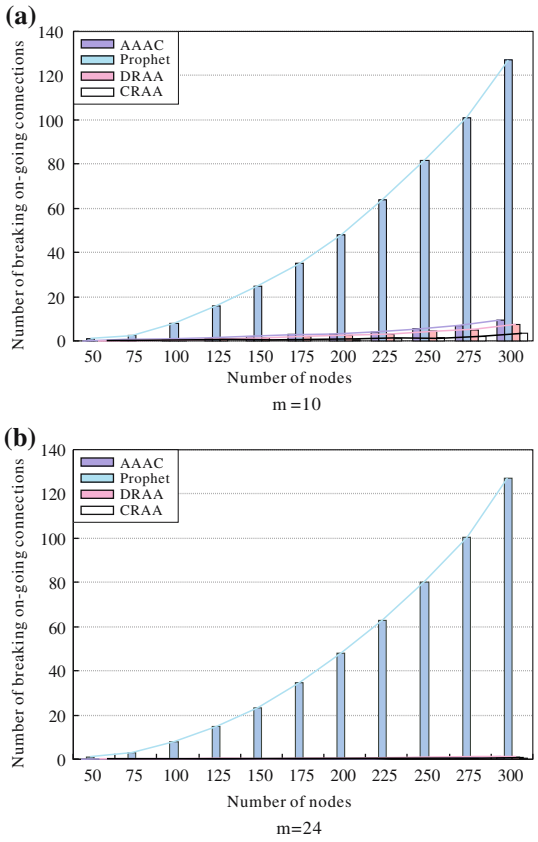

Fig. 21 The number of breaking on-going connections with the varied number of nodes at $m = 10$ and 24

sizes: 2^{10} and 2^{24}, respectively, illustrated in Fig. 21a, b. The buddy system approaches (AAAC, DRAA and CRAA) all perform outstandingly. Since the first node randomly selects an IP address and afterwards allots addresses by dividing an address block into half, the probability of duplicate addressing is rather low during network merging, especially when the address block size is large enough (Fig. 21b), and so is the probability of breaking on-going connections. In Prophet solution, all nodes with the smaller NID must rejoin the network, and all connections should be broken. When it comes to the performance of uninterruptible connections, Prophet performs the worst.

6 Conclusions

This paper proposes two ring-based address autoconfiguration protocols in mobile ad hoc networks. RAA protocols use a logical ring to proceed address allocation and resource management. The ring provides unique address assignment without DAD. Compared with existing address assignment protocols, the DRAA protocol successfully achieves low latency, fewer communication overhead, the evenness and outstanding uninterruptible connection. The DRAA protocol also tolerates one node's invalidity and restores a failed node without help of the holder. Based on the above advantages, the DRAA protocol is suitable for the small and normal scale mobile ad hoc network. The CRAA protocol further achieves low communication overhead and evenness of dynamic address assignment, and restores failed nodes with help of the holder. According to our simulation results, the CRAA protocol shows high efficiency in address allocation as well as in resource management and suitability for the large scale mobile ad hoc network. In the centralized CRAA protocol, the resource management is centrally controlled by holder and its fault tolerance ability is higher enough and similar to the existing protocols. In the decentralized DRAA protocol, the address is requested by unicast communicating to predecessor and the 2nd predecessor. This strategy gets the lower communication time and causes the only single fault tolerance. We can also keep more data and modify our DRAA protocol to increase the fault tolerance ability. This result is not our main concern, because of the communication time and the extra storage will be increased. These two conditions are a tradeoff.

Acknowledgments This research is supported by the National Science Council of the R.O.C. under grants NSC-94-2213-E-194-030 and NSC-95-2221-E-305-008.

References

1. Broch, J., Maltz, D., Johnson, D., Hu, Y. -C., & Jetcheva J. (1998). A performance comparison of multihop wireless ad-hic network routing protocols. In *Proceedings of the fourth annual ACM/IEEE international conference on mobile computing and networking* (*Mobicom'98*). pp. 85–97.
2. Droms, R. (1997). Dynamic host configuration protocol. Internet Engineering Task Force, RFC 2131.
3. Gerla, M., Hong, X., & Pei, G. (2002). Fisheye state routing protocol (FSR) for ad hod networks. Internet

engineering task force, internet draft, draft-ietf-manet-fsr-03.txt, Jun. 2002 (Work in Progress).
4. Jacquet, P., Muhlethaler, P., Clausen, T., Laouiti, A., Qayyum, A., & Viennot, L. (2001). Optimized link state routing protocol for ad hoc networks. In *Proceedings of IEEE international multi topic conference* (*IEEE INMIC*) *2001*. pp. 62–68.
5. Johnson, D., Maltz, D., & Broch, J. (2001). DSR: The Dynamic Source Routing protocol for multihop wireless ad hoc networks. In C. E. Perkins (Ed.), *Ad hoc networking* (pp. 139–172). Wokinghan: Addison- Wesley.
6. Mohsin, M., & Prakash, R. (2002). IP address assignment in a mobile ad hoc network. In *Proceedings of IEEE military communications conference* (*MILCOM 2002*), Anaheim, CA, US, Vol. 2, 856–861.
7. Narten, T., Nordmark, E., & Simpson, W. (1998). Neighbor dscovery for IP Version 6 (IPv6). Internet engineering task force, RFC 2461. (Work in Progress).
8. Nesargi, S., & Prakash, R. (2002). MANETconf: Configuration of hosts in a mobile ad hoc network. In *Proceedings of the twenty-first annual joint conference of the IEEE computer and communications societies* (*INFOCOM 2002*), Vol. 2, 1059–1068.
9. Perkins, C. E., & Bhagwat, P. (1996). DSDV routing over a multihop wireless network of mobile computers. In T. Imielinski, & H. F. Korth, (Eds.), *Mobile computing* (pp. 183–206). Dordrecht: Kluwer
10. Perkins, C. E., & Royer, E. M. (2001). The ad hoc on-demand distance vector routing. In C. E. Perkins (Ed.), *Ad hoc networking* (pp. 173–219). Wokinghan: Addison-Wesley.
11. Perkins, C. E., Malinen, J. T., Wakikawa, R. Belding-Royer, E. M., & Sun, Y. (2001). Ad hoc address Qutoconfiguration. Internet Engineering Task Force, Internet Draft, draft-ietf-manet-autoconf-01.txt (Work in Progress).
12. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Frans Kaashoek, M., Dabek, F., & Balakrishnan, H. (2002). Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*. 149–160.
13. Sun, Y., & Belding-Royer, E. M. (2003). Dynamic address configuration in mobile ad hoc networks. Technical report, computer science department, UCSB.
14. Tayal A. P., & Patnaik, L. M. (2004). An address assignment for the automatic configuration of mobile ad hoc networks. *Personal ubiquitous computer, 8*, (1), 47–54.
15. Thomson S., & Narten T. (1998). IPv6 stateless address autoconfiguration. Internet engineering task force, RFC 2462.
16. Vaidya, N. H. (2002). Weak duplicate address detection in mobile ad hoc networks. In *Proceedings of the 3rd ACM international symposium on mobile ad hoc networking and computing* (*MobiHoc 2002*), Lausanne, Switzerland, pp. 206–216.
17. Weniger, K., & Zitterbart, M. (2002). IPv6 autoconfiguration in large scale mobile ad-hoc networks. In *Proceedings of european wireless 2002*, Florence, Italy, pp. 142–148.
18. Zero Configuration Networking. Retrived from http://www.zeroconf.org/.
19. Zhou, H., Ni, L. M., & Mutka, M. W. (2003) Prophet address allocation for large scale MANETs. In *Proceedings of the twenty-second annual joint conference of the IEEE computer and communications societies ((INFOCOM 2003*) San Francisco, CA, US, Vol. 2, pp. 1304–1311.
20. Haas, Z. J., & Pearlman M. R. (2001). ZRP: a hybrid framework for routing in ad hoc networks. In C. E. Perkins (Ed.), *Ad hoc networking* (pp. 221–253). Wokinghan: Addison-Wesley.

Yuh-Shyan Chen received the B.S. degree in computer science from Tamkang University, Taiwan, Republic of China, in June 1988 and the M.S. and Ph.D. degrees in Computer Science and Information Engineering from the National Central University, Taiwan, Republic of China, in June 1991 and January 1996, respectively. He joined the faculty of Department of Computer Science and Information Engineering at Chung-Hua University, Taiwan, Republic of China, as an associate professor in February 1996. He joined the Department of Computer Science and Information Engineering, National Chung Cheng University in August 2002. Since 2006, he has been a Professor at the Department of Computer Science and Information Engineering, National Taipei University, Taiwan. Dr. Chen served as Co-Editors-in-Chief of International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC), Editorial Board Member of Telecommunication System Journal. He also served as Guest Editor of Telecommunication Systems, special issue on "Wireless Sensor Networks" (2004), and Guest Editor of Journal of Internet Technology, special issue on "Wireless Internet Applications and Systems"' (2002) and special issue on "Wireless Ad Hoc Network and Sensor Networks" (2004). His paper wins the 2001 IEEE 15th ICOIN-15 Best Paper Award. Dr. Chen was a recipient of the 2005 Young Scholar Research Award given by National Chung Cheng University to four young faculty members, 2005. His recent research topics include mobile ad hoc network, wireless sensor network, mobile learning system, and 4G system. Dr. Chen is a member of the IEEE Computer Society, IEICE Society, and Phi Tau Phi Society.

Tsung-Hung Lin is an Assistant Professor of Department of Information Management, Hsing Wu College, Taipei, Taiwan, Republic of China. He received his Ph. D. Degree in Computer Science and Information Engineering from the National Chung Cheng University, Taiwan, Republic of China. His research interests include wireless communication and mobile computing, the next-generation mobile network and system, wireless sensor network, 4G system, and optical computing.

Shih-Min Lin received the BS degree in computer science and Information engineering from the Tamkang, Taiwan, Republic of China, in June 2003 and the MS degree in computer science and information engineering from National Chung Cheng University, Taiwan, Republic of China, in October 2005. His research interest includes mobile ad hoc network.