A Source Tree Reliable Multicast Protocol for Ad-Hoc Networks

Tariq Al-Ahdal, Shamala Subramaniam, Mohamed Othman, and Zuriati Zukarnain
Department of Communication Technology and Networks, University Putra Malaysia, Malaysia.

Abstract: *Multicasting is an essential service for ad-hoc wireless networks. Many reliable multicast schemes were studied in order to reduce packet losses in the network. This paper describes our effort to build a source tree reliable multicast protocol for ad-hoc networks. Source tree reliable multicast protocol provides the delivery of an ordered contiguous sequence of data packets from one sender to many receivers in an ad-hoc network. It is designed to support applications based on bulk data transfer, like files, images and software packages. The core to its support of node mobility, and also what makes the protocol unique, is the dynamic selection of a sub set of 1-hop neighbors from the sender as its forward servers. The key idea behind the selecton of this sub set 1-hop neighbors is to forward the retransmitted lost data packets that are needed by some receivers to achieve a higher throughput and to receive the acknowledgment packet from receivers to avoid the acknowledgment-implosion problem inherent in any reliable multicast scheme. Finally, simulation results show that the protocol has a high delivery ratio and low end-to-end delay compared to ReMHoc protocol.*

Keywords: *Ad-hoc network, forward server, reliable multicast.*

Received January 10, 2007; accepted April 24, 2007

1. Introduction

Mobile Ad-hoc NETwork (MANET) presents a new network model for wireless communication that allows nodes to communicate without the existence of an infrastructure. The nodes that form an ad-hoc network are arranged as a cluster of independent mobile nodes.

The infrastructure-less, self-organizing and mobility features are the internal characterstic of MANETs. However, these also impose challenges, e.g., highly dynamic and unpredictable topological changes, low bandwidth, high error rates and limited power sources, to the protocols and applications for ad-hoc networks. MANETs are being increasingly used for military operations, law enforcement, rescue missions, virtual class rooms, and local area networks. Therefore, such applications depend on multicast operations since they require close collaboration from teams for message exchange. Providing reliable multicast service faces several key challenges in ad-hoc networks as mentioned above. Therefore, there is a need for efficient algorithms that reduce the amount of control and retransmission traffic while multicasting is maintained reliable. Due to the characteristics of wireless ad-hoc, reliable multicast protocols proposed for wired networks may not be suitable to ad-hoc networks. This paper focuses on these two issues: which reliable multicast protocols are more amenable to wireless domain and how effectively they can improve communication reliability over wireless ad-hoc networks.

The remainder of this paper is structured as follows. Section 2 gives an overview of related works, followed by a description of the Source Tree Reliable Multicast protocol (STRM) for ad-hoc networks in section 3. The simulation performance evaluation of STRM is described in section 4. The paper is concluded in section 5.

2. Related Work

A number of reliable multicast protocols have been proposed for ad-hoc networks [5, 8, 9, 11, 12, 13, 15]. These protocols use different approaches to improve packet delivery of multicast routing protocols in ad-hoc networks. One approach is Negative AcKnowledgment (NAK) suppression [9, 11, 13], the receiver is responsible for reliable delivery. Each receiver maintains receiving records and requests repairs via a NAK when errors occur. The problem of this approach is the long end-to-end delay since the sender must wait for the next multicast packet to determine if the previous one is successfully delivered or not. Therefore, it can be applied only when the sender has many packets to be sent. Another way to improve packet delivery is via hierarchical-receiver-oriented approach [5, 8, 12, 15]. A tree for the reliable multicast session is made up of ordinary and special receivers which are called the forwarding regions [10]. Commonly used reliable protocols include the Scalable Reliable Multicast (SRM) [4] and the Reliable Multicast Transport Protocol (RMTP) [7, 14]. SRM is based on an application level framework; the same concept used in Reliable Multicast Protocol for ad-Hoc (ReMHoc) where it is the application's responsibility to guarantee packet sequencing. ReMHoc [11] is a receiver-initiated NAK-based reliable multicast protocol. This protocol uses random timer-based feedback suppression in order to avoid NAK and retransmission implosion. It has also incorporated a 'heartbeat' timer, which is used to keep peer members updated on multicast packets. But the *repeat* and *request* timers depend on the numbers of hops between nodes which is not only in accurate in a mobile scenario but also causes extra overhead and more delays.

Some approaches to provide reliable multicasting in wireless ad-hoc networks include Active Reliable Multicast Protocol with Intermediate Node Support (ARMPIS) [15] and Reliable On-Demand Multicast Routing Protocol (RODMRP) [12]. ARMPIS distributes multicast message cache and retransmission tasks among intermediate nodes to offer a scalable reliable multicasting. On the other hand RODMRP is an extension to the ODMRP [6], the protocol designed for multicast and unicast routing. In essence, RODMRP leverages the information propagated by ODMRP to determine which the nodes are directly downstream from the sender and forwarding nodes. Once this is determined, each outstanding data packet in the transmission window is unicast to each downstream neighbor in the mesh in a *round-robin* fashion. This peculiar windowing mechanism has the potential for multicast sessions with long delays, because the whole session can be slowed down by intermediate nodes that are overloaded or just slow to respond to its upstream sender.

The Reliable Multicast Algorithm (RMA) [5] is an ACK-based reliable multicast protocol. Unlike other reliable multicast protocols that assume underlying multicast protocols, RMA is a multicast protocol supporting reliable transmission using *ack* from receivers to sources. The developers in this protocol assume that the sources have the full knowledge of group membership via *join* or *ack* messages. RMA is a sender-initiated multicast protocol. The sender guarantees retransmissions of lost packets. RMA also uses a novel link cost criterion, *link-lifetime*, to improve reliability. Choosing a path with longer life time plays a vital role in an unstable environment as a MANET. The sender also favors paths composed of more group members over those with fewer members. Thus more aggregation can be implemented in a single message, resulting in less message forwarding and less bandwidth usage. However, in RMA all the receivers must send *ack*s back to the sender for received data packets. This adds burden to the sender and will cause *feedback implosion* when the group size grows.

Another solution for implementing reliable multicast in mobile ad-hoc environments is proposed in [13] and later on is enhanced in [9]. The latter does however make uses of both source-oriented and local recovery mechanisms. The source-oriented component works the same as RALM [13], which is omitted here. The local recovery is the major contribution of ReAct [9]. Local recovery occurs right after the receiver detects a lost packet. Local recovery mechanism considerably impacts the overall performance of RALM. In particular the scheme works effectively when packet losses are due to random errors, e.g., mobility and link errors. Local recovery gets missing packets faster than source-oriented retransmission, reduces the burden/congestion at the source, and alleviates potential feedback implosion problems. However, worst case scenarios exist for ReAct when local recovery frequently fails and source recovery is triggered all the time. When this happens, mostly possible in high mobility, longer delays and low throughput dominate the data delivery, leading to serious degradation of network performance.

Epidemic-based reliable and adaptive multicast (EraMobile) [8] is epidemic-based reliable multicast for mobile ad-hoc networks. EraMobile utilizes an epidemic-based method in multicast operation to cope with dynamic and unpredictable topology changes arising from the mobility. This protocol provides fully reliable multicast data delivery with minimal network overhead even in the adverse network conditions. EraMobile does not employ a separate mechanism to have the information of neighbors around a node. Instead, it utilizes the periodic gossip broadcasts and request messages received.

Observing that providing reliability of data delivery is a NP complete problem [3]. Moreover, retransmission is the best way to recover lost packets. Therefore the predictability of choosing a sub set of 1-hop neighbors from the sender as a forward for lost packets will be reducing the burden in the sender. The proposed algorithms aim at achieving this. In the following section, the new proposed reliable multicast protocol for ad-hoc networks will be explained.

3. Protocol Description

STRM is designed to provide delivery of an ordered contiguous sequence of data packets from one sender to many receivers in an ad-hoc network. It is designed to support applications based on bulk data transfer, like files, images and software packages. Given the complexity of the subject, STRM does not have any congestion control mechanism. The protocol assumes the existence of a multicast mesh, provided by an underlying multicast routing protocol called ODMRP [6]. It could, however, be adapted to work with a multicast tree as well. STRM does not perform any repair for partitioned meshes or trees, and focuses entirely on the reliable transmission of application data in the presence of network mobility.

The STRM protocol is based on method of local recovery, periodic regular *ack* being sent by the receiver to its FSs, and dynamic re-definition of the FSs. This is the mechanism called Selection Forward

Server Process (SFSP), and is discussed in detail later in this paper. This mechanism makes STRM different from other protocols. STRM has three entities:

- *Sender*: the sender has a controller component which decides whether the sender should transmit new packets; retransmit lost packet or send messages to advertise itself as an *ack* processor, process *ack*s (status) that come from receivers and update relevant data structures according to them. The sender also has another component which decides a set of multicast a FSs to all multicast group.
- *Receivers*: each receiver entity has a controller component which decides whether the receiver should receive data packets or send an ACK packet.
- *Forward Servers (FS)*: these entities are a combination of the sender entity and receiver entity.

3.1. STRM Details

The idea behind STRM is simple. In essence, the sender breaks the data to be transmitted into fixed-size data packets, with the exception of the last one. The sender assigns each packet a sequence number starting from 0. Each receiver periodically sends ACK packet to its parent to inform it about the packets that the receiver has received correctly and those packets that are needed by the receiver. The sender and FS nodes collect ACK packets, or retransmission requests, but do not wait for all children to respond, since adjacent nodes may move away from each other at any time. At regular time intervals, the sender and intermediate nodes respond to retransmission requests, but only to the ones available in the retransmission queue.

The protocol uses windowing mechanisms for its operation. The sender uses a transmission window (W_s) to keep track of data packets that have been sent but are not yet acknowledged, and also packets that have not been sent at all. The sender also has a memory window (W_m), which can be seen as a superset of the transmission window. The memory window stores a limited number of packets that have recently been acknowledged, with the purpose of responding to retransmission requests coming from late receivers.

For receivers, the protocol defines the receive window (W_r), which stores data packets that have been received up to a given instant but could not be delivered to the application layer because do not form a contiguous sequence. The ACK packets contain the lower end of the receive window and a bitmap indicating the packets that have been received successfully and the ones that have not.

3.2. Reliable Transmission and Acknowledgments

The sender transmits new data packets at regular interval defined by a configuration parameter, T_{send}. The number of packets sent at every such interval depends on the available room in the transmission window. The sender can transmit at most one full window of packets (W_s) during T_{send} period, thereby limiting the sender maximum transmission rate to:

$$W_s * Packet\_size / T_{send} \tag{1}$$

where W_s is the size of the transmission window.

Receivers send ACK packets periodically every time of ACK, T_{ack}, to their FSs, indicating the status of their receive window. The status information sent on *ack* packets consists of the sequence number of the left bound *Wr_lb* or *lb* of the receive window and a bitmap, *B*, of 0's and 1's. The last bit of this bitmap contains the sequence number, *N*, of the lowest unstable packet. The lowest unstable packet is the packet with sequence number lower then this packet has been received correctly. The 1's indicates that the corresponding packets have been received correctly and the 0's indicates that the corresponding packets need to be retransmitted. Based on these status messages, the sender and the FSs determine which packets need to be retransmitted.

The bitmap consists of W_r bits (size of the receiver window) to record the existence of the correctly received packet stored in their buffers. For example, if the ACK contains *N*=19 and *B*=01110101. This indicates that the receiver has correctly received packets with sequence numbers less than 19 and that it is requesting the retransmission of packets 19, 23 and 25 as indicated by zeros present in the bitmap *B*.

Both sender and FS have a retransmission queue, Q_{retx}, where each entry consists of a sequence number of a packet and list of receiver nodes asking for that packet to be resent. The retransmission queue is examined every T_{retx} by the sender and FS nodes. If the queue is not empty, each packet is multicast to the requesting nodes if the number of nodes exceeds a threshold; otherwise the packet is unicast only to the requesting nodes.

3.3. Mobility

Mobility can cause the multicast mesh to become partitioned. This will certainly cause some receivers to loose connectivity with the rest of the multicast group. STRM provides means for the receiver to attempt recovery from that situation. The lack of connectivity can be detected by a receiver when, for a period of time, the data stream stops and the final data packet has not been received.

ODMRP has a mechanism to overcome situations of mesh partitioning. While the sender has packets to send, ODMRP expected to periodically exchange JOIN DATA, JOIN TABLE messages to avoid mesh partitioning. These periodic transmissions refresh the membership information and update the route.

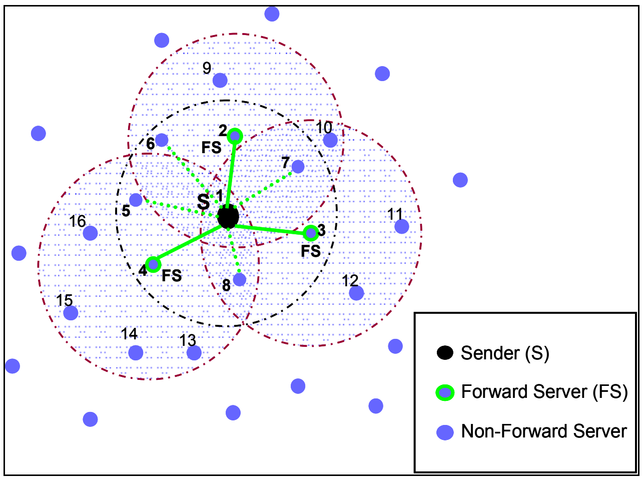

Figure 1. A sample network where the sender S uses the SFSP to select its FSs.

The most relevant aspect in STRM's design for the support of mobility is the dynamic definition of the FSs. In STRM, the sender selects a subset of 1-hop neighbours in the mesh topology as FSs to forward multicast data packet and receives the *ack* packet from receivers. The selection is based on a neighbor utility approach, U_n (equation 2). The neighbor utility U_n for a node i equals to the number of unallocated nodes in two hops pools that are neighbours of node i divided by the total number of neighbours of node i. The selected FSs must cover all the nodes within 2 hops of the sender. The sender waits for a predefined duration to receive *ack*s from its FSs. If the sender does not receive all *ack*s from its FSs in sending time of *ack*, it assumes that a transmission failure has happened for this multicast and that the packet needs to be resent. If the sender fails to receive *ack*s from all its selected FSs after sending the packet a threshold number of times, the sender assumes the FS that do not reply are out of its transmission range and it stops further attempts.

$$U_n(i) = \frac{\text{unallocated two hop neighbours of node}_i}{\text{total two hop neighbours of node}_i} \quad (2)$$

We apply the following extension to improve the performance of the algorithm: When a sender u fails to receive an *ack* from its FS v after a maximum number of retries, u reselects an alternative FS to cover the set which is supposed to be covered by v.

The algorithm that requires only the selected FSs send *ack*s, which is commonly used for nodes sending NAKs to inform the sender of the missing packet, can avoid the ACK implosion problem.

3.4. SFSP

In selecting forward server process SFSP, we use $N_k(v)$ to represent the neighbor set of v, where nodes in the set are not further than *k_hops* from v. $N_k(v)$ includes v itself. ($N_1(v)$, 1-hop neighbour set, can be simply represented as $N(v)$) Neighbouring nodes exchange their 1-hop neighbor set information; therefore, each node v has its 2-hop neighbor set information $N_2(v)$. The SFSP executes at the sender to determine its own forward server set: A sender S selects its forward server set from its 1-hop neighbor set $N(S)$ to cover all the nodes in its 2-hop neighbor set $N_2(S)$. Therefore, each node v in $N(S)$ can be one of two cases: (1) v is a FS, it will not discard the packet from their buffer until all their children have received the multicast packet and retransmitted lost packets by multicasting them only to the requesting receivers in local group of the FS. (2) v is not a FS, if v is a receiver node, it will reply an ACK to the FS, else the node will discard the packet. The FS are selected based on the following greedy algorithm.

```
$H_1$ = 1-hop neighbor(s)
$H_2$ = 2-hop neighbor(s)
the FS is initialized to be empty at the sender buffer
FS = nodes in $H_1$ with unique neighbor(s) in $H_2$
$H_1 = H_1$ - FS
$H_2 = H_2$ - N(FS)
while $H_1 \neq \phi$ or $H_2 \neq \phi$
    for each node in $H_1$
        calculate its $U_n$
    endfor
    n = highest utility node from $H_1$
    $H_2 = H_2$ - N(n)
    FS = FS + n
    $H_1 = H_1$ - n
    remove nodes from $H_1$ with $U_n = 0$
endwhile
return FS
```

Figure 2. Algorithm SFSP.

In the sample network topology shown in Figure 1, $N(I)$ = {1, 2, 3, 4, 5, 6, 7, 8} and $N_2(I)$ = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}. When using the SFSP algorithm in Figure 2, sender node S selects nodes 2, 3 and 4 as its FS nodes. In the algorithm in Figure 2 the sender uses this algorithm to generate a pool of one hop (H_1) and two hop (H_2) neighboring nodes. All nodes that are neighbors of the previous multicasting are removed from both pools. Nodes in H_1 with unique neighbors in H_2 are removed from H_1 and added to the set of FSs. Thus all one hop neighbors of the FS are removed from H_2. The remaining nodes in H_1 with no unique neighbors are assigned a neighbor utility (U_n) in Equation 1. Then an allocation occurs, which adds the node in H_1 with the highest utility to the set of FSs and removes its neighbors from H_2. After that the neighbor utility for the remaining nodes is revised. This continues until H_1 or H_2 is an empty set. The set of chosen FSs is then attached to the multicast message at the sender. Nodes which are not in the attached FS set are inhibited from re-multicasting.

4. Performance Evaluation

This section describes the simulation conducted to evaluate the performance of STRM. We investigate the behavior of STRM and compare it with a protocol which uses a NAK-based buffer management scheme, denoted as ReMHoc in the rest of this paper, where each receiver node multicasts a NAK to entire group whenever it detects a packet loss, as in [11], the comparing depend on delivery guarantee and bandwidth consumption (control overhead).

4.1. Description of the Simulation Program

In order to evaluate the performance of the STRM protocol, we developed the simulation program using Visual C++ language. We assume that all nodes and links work properly and none of them fails during the simulation time. For each run, the simulation used different uniform distribution to generate a random network topology. The simulation program is run 10 times for the same configuration parameters. The result is taken as the average among these iterations in order to have a stable result.

The simulator is developed based on event-driven. The main events in the simulator are SEND, REQUEST and RETRANSMISSION. All these events are scheduled per sender and receivers; the sender periodically sends a window of data packets. The first data packet of this window has the value of the number of the packets the sender will send in the current window and sequence numbers. Every receiver upon receiving this value waits until it receives these packets and sends with the next *ack* packet to its FS the time it received the last packet of these data packets. A more detailed, discussion and validation of the results can be found in [1].

4.2. Simulation Environment Model

In the simulation, we focus on the impact of nodal mobility on the performance of the STRM protocols. We consider a 700x700 meters mesh topology in which nodes are roaming in the mesh during the simulation. We generate a set of *N* mobile nodes, initially located at *X*, *Y* coordinates randomly selected in the mesh topology, where *N* varies from 1 to 30. We assume that there is only one sender, randomly selected from the *N* mobiles. Each node has a transmission range of 250 meters, within which mobile nodes can directly communicate with one another. A multicast delivery tree is rooted at the sender and spans over all other nodes. The sender generates data packets at a constant rate of two packets per second. The size of the implementation window is 4 packets per window.

Each node moves according to the mobility model defined in [3]. Initially, each node randomly selects a location in a 700-by-700 meters mesh topology. On expiry of a pause time, a node moves to another randomly selected coordinate in the mesh at a speed uniformly distributed between 0 and 25 m/s. Once having reached the destination, the node pauses again for another pause time. Then it selects another destination and speed and moves again. We use six different pause times in the simulation: 1, 3, 6, 10, 15, and 20s. The shorter the pause time, the higher the mobility.

We also compare the performance of STRM with the pure Source-Based Scheme (PSB), where only the original sender in STRM allows the retransmission of packets in response to NAKs. The aim of this comparison is to study the effectiveness of STRM's local recovery mechanism. We compare STRM with ReMHoc both running on top of the ODMRP [6]. ReMHoc is selected because it employs error control, instead of congestion control, to achieve reliable delivery.

4.3. Simulation Results

This experiment is conducted to compare between STRM, PSB, ReMHoc, and ODMRP protocols. A parameter called the reliable delivery ratio is defined as the fraction of packets successfully (or reliably) delivered to all receivers over the total number of packets sent. This number represents the routing effectiveness of a protocol. The larger the delivery ratio, the larger the number of lost packets that can be recovered, and the better the performance. We can see from Figure 3 that as speed increases, the routing effectiveness of ODMRP degrades rapidly compared to STRM and ReMHoc protocols. STRM has very high delivery ratios of over 98% regardless of the speed. While ReMHoc and PSB degrade rapidly when the speed is more then 15 m/s. As the routes are reconstructed in advance of topology changes, most data are delivered to multicast receivers without being dropped.

Figure 3 demonstrates how STRM is able to achieve a greater reliability of data delivery. The statistics were collected with a session size of 30 nodes. Figure 4 was achieved under similar networks considerations. It demonstrates the percentage of requests packets, which is the ratio between the number of requests for lost packets transmitted by receivers and the total number of original data packets transmitted by the sender. Duplicate request messages are taken into account in these measurements.

Then, mean values are calculated for each simulation. The results show that, as the mobility speed scale up, the number of control messages received by group members during loss recovery increases linearly for ReMHoc protocols. The costs remain almost constant for STRM.

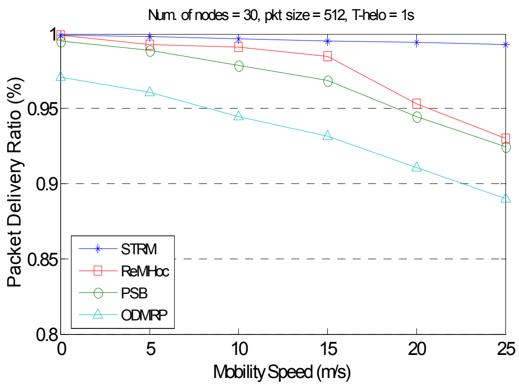

Figure 3. Effect of packet delivery ratio with mobility speed.

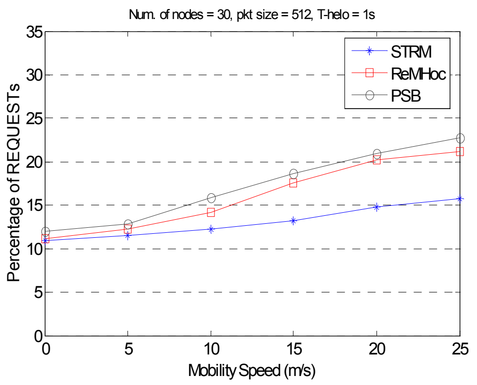

Figure 4. Percentage of requests with mobility speed.

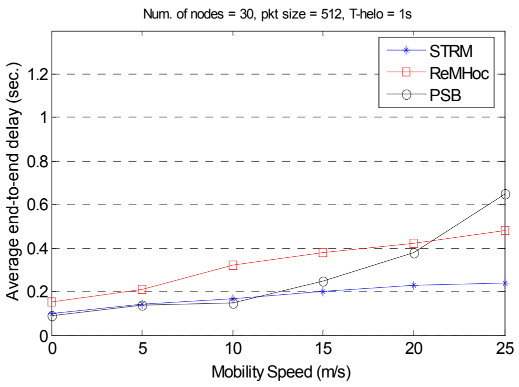

Figure 5. Average end-to-end delay with mobility speed.

Figure 5 demonstrates the average end-to-end delay, which is calculated as the average difference between the time each data packet is transmitted by a sender and the time it is received by a receiver entity, and then averaged over the total number of receivers. In the figure, ReMHoc has a larger delay than STRM and PSB schemes due to a high control overhead and thus large queuing delay. STRM has shorter delays than PSB, and this difference becomes more obvious as mobility speed increases.

Figure 6 shows the percentage of retransmission packets, which is the ratio between the number of retransmission packets transmitted by retransmission group member (both sender and FS entities) and the total number of original data packets transmitted by the sender entities. In ReMHoc protocol, each member negatively acknowledges packet losses to the group receivers. Any lost packets can be retransmitted by any member in the session that has the losses packet after periods of time. Thus, the ReMHoc protocol has a higher delay then others.

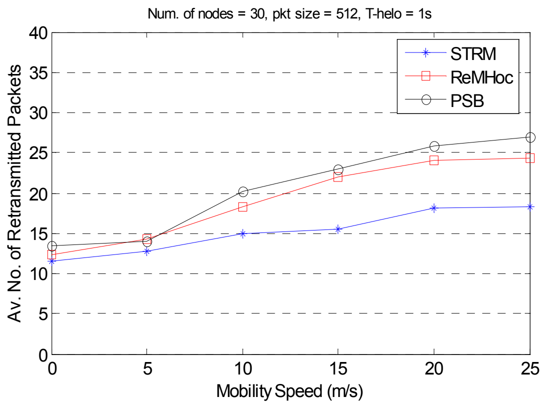

Figure 6. Average no. of retransmited packets with mobility speed.

Figure 7 shows the effect of increasing the mobility speed with the average recovery latency. The average difference between the time at which a receiver entity detects each missing packet is received at the receiver entity, and then the difference is averaged over the total number of receiver entities.

Figure 8 calculates the overhead percentage, which is the ratio between the total protocol overhead (i.e., the sum of all ACKs, redundant data packets either retransmission or duplicate data packets) received by each receiver entity and the total number of original data packets transmitted by sender entities.

Figure 9 shows the numbers of ACKs packets that have been sent from the multicast group nodes to the sender and FSs in the STRM and ReMHoc protocols. As the Figure shows the number of ACK packets that STRM protocol received from its receiver nodes is greater than ReMHoc protocol, which means the FS nodes face more overhead to manage this ACK packets because the receiver nodes in ReMHoc protocol depend on the NAK scheme.

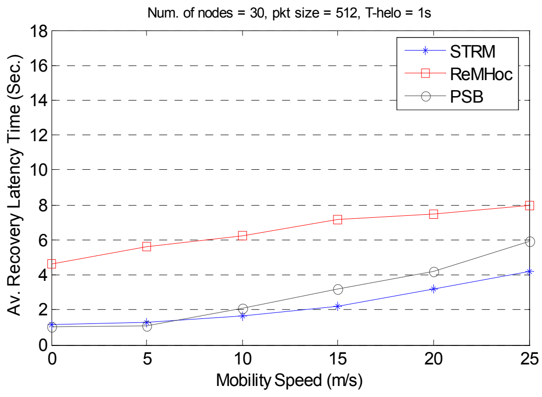

Figure 7. Average recovery latency time with mobility speed.

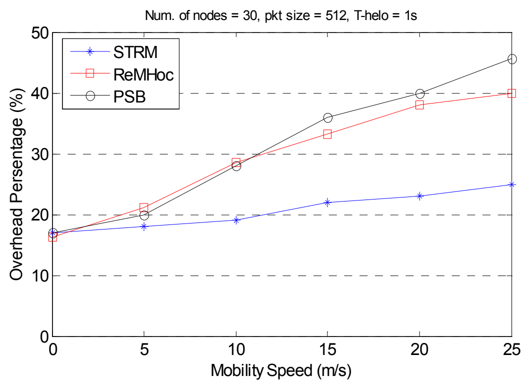

Figure 8. Overhead percentage with mobility speed.

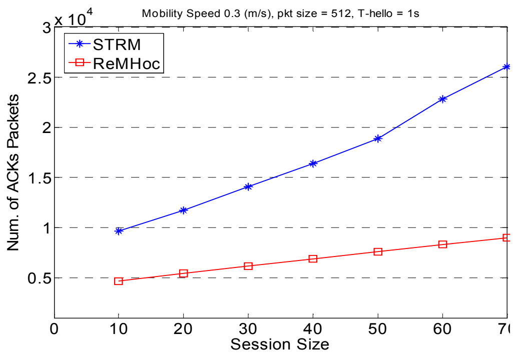

Figure 9. Num. of ACKs packet with session size.

5. Conclusion

In this paper, we introduced a novel reliable multicast transport protocol, STRM, to provide a high degree of reliability in multi hop ad-hoc networks. We study the impact of local recovery by creating FSs nodes and regular ACK for STRM. The better performance of STRM can be traced to the following two reasons. First, utilizing the local recovery technique shortens the delay as the chance of congestion is reduced. Second, adjusting to network mobility via receiving beacon messages from neighbors yields faster convergence. In PSB, a neighbor displacement is noticed only after a packet is sent explicitly to a node. Also the network reacts if an ACK is not received. Consequently, this increases packet delay since the packet must wait until a new route is established. In our future work, since there is a large overhead in forward server nodes, their buffers should be managed in an efficient manner. We will use ordered ACK scheme to provide an efficient way to discard packets from forward server's buffers. In this scheme each receiver needs to send a NAK to its FS every time they detect a packet loss. The nodes also will send an ACK packet to its FS. This packet will act as an implicit ACK for the previous packets that is received in the sending period.

References

[1] Alahdal T., Shamala S., Othman M., Zukarnain Z., and Alsaih A., "A Simulation Tool for Reliable Multicast Transport Protocol Analysis in Ad-Hoc Network," *in Proceedings of the International Conference on Advanced Technologies on Telecommunications and Control Engineering (ATTCE'2006)*, INTI Colleges Malaysia, vol. 1, no. 3, August 28-29, 2006.

[2] Bettstetter C., Resta G., and Santi P., "The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad-Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, pp. 25-39, 2003.

[3] Chlamtac M. and Liu J., "Mobile Ad-Hoc Networking: Imperatives and Challenges," *Ad-Hoc Network*, vol. 1, no. 1, January 2003.

[4] Floyd S., Jacobson V., Liu C., McCanne S., and Zhang L., "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 784-803, December 1997.

[5] Gopalsamy T., Singhal M., Panda D., and Sadayappan P., "A Reliable Multicast Algorithm for Mobile Ad-Hoc Networks," *in Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems*, Vienna, Austria, pp. 563-570, July 2002.

[6] Lee S., Gerla M., and Chiang C., "On-Demand Multicast Routing Protocol," *in Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'99)*, pp. 1298-1304, 1999.

[7] Lin J. and Paul S., "RMTP: A Reliable Multicast Transport Protocol," *in Proceedings of the IEEE INFOCOM'96*, pp. 1414-1424, 1996.

[8] Özkasap Ö., Genç Z., and Atsan E., "Epidemic-Based Approaches for Reliable Multicast in Mobile Ad-Hoc Networks," *ACM SIGOPS*, vol. 40, no. 3, pp. 73-79, July 2006.

[9] Rajendran V., Obraczka K., Yi Y., Lee S., Tang K., and Gerla M., "Combining Source and Localized Recovery to Achieve Reliable Multicast in Multi-Hop Ad-hoc Networks," *in Proceedings of the Networking' 04*, pp. 112-124, May 2004.

[10] Sadok D., Cordeiro C., and Kelner J., "A Reliable Subcasting Protocol for Wireless Environments," *in Proceedings of the 2nd International Conference Mobile and Wireless Communications Networks*, Paris, France, pp.174-185, 2000.

[11] Sobeih A. and Fahmy A., "ReMHoc: A Reliable Multicast Protocol for Wireless Mobile Multi Hop Ad-Hoc Networks," *in Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC'2004)*, USA, pp. 146-151, 2004.

[12] Tang K. and Gerla M., "Reliable Multicast of the On-Demand Multicast Routing Protocol," *in*

Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics, Orlando, FL, pp. 1009-1014, July 22-25, 2001.

[13] Tang K., Obraczka K., Lee S., and Gerla M., "Congestion Controlled Adaptive Lightweight Multicast in Wireless Mobile Ad-Hoc Networks," *in Proceedings of the IEEE Symposium on Computers and Communications (ISCC)* pp. 531 - 540, July 2002.

[14] Whetten B. and Taskale G., "An Overview of Reliable Multicast Transport Protocol II," *IEEE Network Magazine*, vol. 14, no. 1, pp. 37-47, 2000.

[15] Wu S. and Bonnet C., "ARMPIS: An Active Reliable Multicasting Protocol for Ad-Hoc Networks," *in Proceedings of the 8th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI'2004)*, Orlando, Florida, USA, 2004.

Tariq Al-Ahdal is a PhD Student of the Department of Communication Technology and Networks, Faculty of Computer Science and Information Technology, University Putra Malaysia. His research area focuses on reliable multicasting protocol, especially, congestion control and reliable protocol for multicast protocol. He is also doing research on reliable multicast routing protocol for ad-hoc networks.

Shamala Subramaniam received her BS, MS, and PhD degrees in dynamic traffic scheduling and resource reservation algorithms from University Putra Malaysia, Malaysia, in 1996, 1999 and 2002, respectively. Currently, she is an assistant professor in the Department of Communication Technology and Networks faculty of Computer Science and Information Technology, University Putra Malaysia, Malaysia. Her research interests include computer networks, simulation and modeling, scheduling, and real time systems.

Mohamed Othman received his PhD from National University of Malaysia with distinction (Best PhD thesis in 2000 awarded by Sime Darby Malaysia and Malaysian Mathematical Society). In 2002 and 2003, he received gold medal award for research development exhibition. His main research interests are in the field of parallel and distributed algorithms, high speed computer networks, network management (security and traffic monitoring), and scientific computing. He already published several journal papers. He is also an associate researcher and coordinator of high speed machine at institute of mathematical science, UPM.

Zuriati Zukarnain is head of Department of Communication Technology and Networks, UPM. She received the BSEd degree (Hons) in physics from UPM in 1997, MS in information technology from UPM in 2000, PhD degree in quantum computation and information from University of Bradford, United Kingdom, 2006. Her research interests are quantum computation, quantum information and communication (entanglement, teleportation), computer networks, and distributed computing.