COMMUNICATION SUPPORT FOR TIGHTLY-COUPLED DISTRIBUTED MOBILE APPLICATIONS

IRINA GERASIMOV and ROBERT SIMON

Laboratory for Cooperative Work Environments
Department of Computer Science
George Mason University
Fairfax, VA, 22030
{igerasim, simon}@cs.gmu.edu

Abstract: One of the major challenges faced by designers of ad hoc network systems is supporting the deployment of applications that require tightly constrained communication support. Examples of this application class include emergency response systems, military systems and mobile or sensor-based scientific applications. This paper describes QoS-AODV, an integrated End-to-End Delay (EED) bounded route discovery and bandwidth reservation protocol. QoS-AODV is designed to operate within a Time Division Multiple Access (TDMA) network. Unlike other path finding protocols that ignore the impact of the data link layer, QoS-AODV incorporates slot scheduling information to ensure that bandwidth reserved and EED requirements satisfied as well. To minimize EED, QoS-AODV uses an efficient heuristic algorithm for TDMA slot assignment that in addition provides zero delay jitter. QoS-AODV is an enhanced version of the Ad hoc On-demand Distance Vector routing protocol, and is therefore compatible with proposed route discover and maintenance techniques. In order to test the effectiveness of our protocol we implemented a version of QoS-AODV, along with several QoS enable variations, in the ns-2 simulator. Our experiments showed the QoS-AODV significantly improved the probability of being able to find an end-to-end QoS path.

Keywords: Ad Hoc Network QoS, Wireless TDMA

1. INTRODUCTION

A tightly-coupled distributed application can be characterized as an application class that requires stringent synchronization of all computational and communication activities. Examples of such applications include real-time collaboration systems and fine-grained parallel and distributed scientific computing applications. Supporting these performance requirements presents significant challenges for distributed system designers. As a result, scheduling and task assignment techniques for this application class have long been a topic of research. The problem has been considered from a variety of infrastructure options, including Networks of Workstation (NoWs), cluster computing, and parallel computer architectures. Many of the proposed algorithms and scheduling approaches have assumed the presence of a reliable communication infrastructure whose performance can be predicted in advance, even when individual nodes or workstations are communicating over a multi-hop network.

Recently, however, there has been the widespread interest in and use of mobile and wireless computing systems. It is likely that these systems will be required for many types of applications previously deployed over NOW-like environments. For instance, wireless multi-hop systems can be expected to provide underlying system support for emergency responders, within a classroom or conference setting, or in any environment that does not or cannot have wired communication support. The applications run in this environment might include real-time multimedia collaboration software, computationally intensive algorithms scientific programs that need input from multiple sensors, etc. In order to deploy many types of tightly-coupled applications within such an environment, in is necessary to provide communication channels that have predictable performance, and that exhibit minimal variation in communication performance. This activity is typically cast in terms of a network Quality-of-service (QoS) problem.

Tightly-coupled distributed applications require predictable system performance for three different QoS parameters - bandwidth, end-to-end delay (EED) and jitter. These parameters can be characterized in terms of average or worst case conditions. It is also desirable to minimize the number of hops in a route. This is true for all networks, but is particularly important in an ad hoc environment, where links themselves may break due to the node mobility.

This paper presents the design and simulation analysis of a novel technique for communication support of tightly-coupled parallel and distributed applications in a mobile ad hoc computing environment. An ad hoc network is a distributed system of wireless mobile computers that can temporarily form a multi-hop network without the aid of a fixed infrastructure. Due to unpredictable and potentially rapid changes in route and bandwidth availability, significant challenges need to be addressed before QoS techniques can be deployed in ad hoc environments. Unlike much of the previous work in ad hoc QoS networking, we focus on communication support satisfying all three QoS parameters. Our approach combines the Ad Hoc On-demand Distance Vector (AODV), a network-layer path finding technique, with a variety of data link scheduling approaches. The combination of distributed EED calculation and data link scheduling provides a unique mechanism for QoS path finding that can be used to complement a base ad hoc routing protocol. Our proposed QoS methodology maximizes the likelihood of finding routes that can support the traffic requirements of tightly-coupled applications in ad hoc networks.

At the data link level, a fundamental problem for wireless systems is scheduling different node transmissions such that they do not interfere with each other. One manifestation of this problem is the "hidden terminal" effect. Here, two nodes, A and B, want to transmit at the same time, potentially to the same node C. The problem is that nodes A and B cannot hear each other's transmission because they are out of each other's range. However, their transmissions collide at node C. There have been a variety of protocols and techniques used for the scheduling transmission and resolving contention issues at the wireless data link layer, including 802.11 and Code Division Multiple Access (CDMA). In our work, we assume that the underlying data link protocol uses a Time Division Multiple Access (TDMA) structure [Stallings 2002]. In TDMA, the time axis is divided into a sequence of frames. Nodes communicate with each other by reserving time slots within each frame. By coordinating the reservations of all participating nodes collisions are avoided and end-to-end QoS can be achieved. It is worth noting that TDMA schemes can be overlaid on both 802.11 and CDMA environments.

Our integrated approach is called QoS-AODV. As indicated above, it combines on-demand routing and path finding with a TDMA scheduling and data link layer resource reservation approach. Our protocol is quite flexible, in the sense that it can integrate several different types of slot scheduling algorithms, including a jitter-free technique that is particularly suited for

tightly-coupled distributed applications. Our protocol sets up a Virtual Circuit (VC) between each sender and receiver, such that the VC satisfies the applications QoS requirements.

Our environment is assumed to consist of a small to medium ad hoc network, consisting of anywhere from 10 to 30 nodes. Notice that this figure is in terms of potential network layer routing hops. The actual number of user-level mobile nodes may be much larger, and many user stations may be "clustered" into a single network layer node. This is typical of a scenario involving an emergency response, or a classroom setting, where the longest network layer routing path is no more than 5 to 10 hops.

In order to test the performance of our protocol we have implemented QoS-AODV in the ns-2 simulator. We evaluated the performance of our slot assignment mechanism, called Zero Jitter with Phase Alignment (ZJPA), by comparing it with other three algorithms: FIFO, Random and Zero Jitter (ZJ). We used the Call Acceptance Ratio (CAR) as a performance metric. Our results showed that CAR for our ZJPA algorithm does not change for different EED requirements. In contrast to that, CAR for FIFO and Random algorithms for EED as low as a half of TDMA frame drops almost to zero and CAR for ZJ drops by 20%.

As a further point of comparison we also evaluated a simpler version of QoS-AODV that lacks our QoS path finding mechanism. Our results indicate that for a variety of mobility scenarios QoS-AODV substantially increases the likelihood of finding of suitable path that meets all named QoS requirements.

The paper proceeds as follows. Section 2 provides some background and shows related work in this area. Section 3 shows our path finding approach, which includes the calculation of the end-to-end delay. Section 4 describes our approach to scheduling for jitter minimization. Section 5 shows the results of our performance evaluation, and Section 6 offers some conclusions.

2. BACKGROUND AND RELATED WORK

Algorithms and scheduling methodologies for supporting tightly-coupled distributed applications have typically assumed the existence of a communication network whose performance is predictable [Bevilacqua 1999; Lumetta et al 1997]. In particular, the underlying network is assumed to offer a stable routing pattern, and relatively jitter-free packet delivery. However, these assumptions break down in a mobile wireless environment. While there has been a tremendous

amount of recent research for communication support within mobile systems, this work has typically not focused on the need to provide jitter-free QoS in an ad hoc environment for tightly-coupled applications. There have been some efforts at exploring issues of cluster computing in a wireless environment, although not in an ad hoc system [Zheng et al, 1999].

2.1. QoS support in an ad hoc environment

Supporting tightly-coupled distributed application entails finding routes that satisfy four major QoS metrics: bandwidth requirements, EED, delay jitter and the number of in the path VC. One research area takes well-known Internet QoS models such as Integrated or Differentiated Services and applies them to network of mobile hosts [Xiao et al 2000; Cansever et al 1999]. Because these approaches are entirely independent of the data link layer, and because the focus of this work is on large-scale networks, they do not directly provide support for tightly-coupled applications.

For QOS routing algorithms based on non-TDMA MAC transmission protocols, the meanings of the above QoS metrics are quite transparent. The bandwidth requirement is the quantity of bandwidth that has to be reserved at each node. EED is the average packet transmission time from the source to the destination nodes that include media transmission delay and queuing and processing delay at each node in the path. Delay jitter is the difference in minimum and maximum time encountered between consecutive packet arrivals. Path cost is the number of hops along VC and it is tightly coupled with EED. For non-TDMA MAC based QoS algorithms EED metric becomes a path cost, since EED is directly proportional to the number of hops along the path.

An example of QoS routing with non-TDMA MAC layer is presented in [Chen and Nahrstedt, 1999]. This kind of QoS routing assumes that the MAC layer provides the necessary mechanisms for QoS guarantees. The approach is based on distributed on-demand path search, which uses known link bandwidth between nodes, and finds paths that have the required bandwidth, EED, and number of hops. While this approach is fully distributed and scalable, the lack of mechanisms to control quantitative measure of bandwidth information from existing MAC layers is a drawback.

For QOS routing algorithms based on TDMA MAC layer, the meanings of the considered QOS requirements are transformed due to the underlying TDMA architecture, consisting of TDMA time frames and time slots. The bandwidth requirement becomes a number of time slots that should be reserved on each node along VC. EED is defined as the difference between the time the packet was issued by application at the source node and the time the packet was accepted by the application at the destination node. Delay jitter is the difference in minimum and maximum number of time slots between two sequential packet arrivals to the destination node. Finally, the path cost definition is the same and it is expressed as the number of hops of the VC. It is important to note that in TDMA MAC-based routing the EED may not be directly proportional to the hop count metric. While EED is still a function of number of hops in the path, it is also heavily depends on TDMA time slot scheduling on each node along that path. Since TDMA time frames can be of the length of tenth of the second and can have hundreds of time slots, some TDMA schedules lead to delays that close to one time frame per hop. This kind of slot scheduling delay can easily overcome all other kinds of delays. This is a strong reason to perform EED calculations that results from MAC TDMA scheduling of VC nodes.

Examples of QoS implementation over TDMA-based ad hoc networks can be found in [Lin and Liu 1999; Lin and Liu 2000; Ho and Liu 2000; Gerasimov and Simon 2002]. In those approaches the bandwidth calculation and allocation is done by reservation of TDMA or Code Division Multiple Access (CDMA) time slots. While the mechanisms of QoS routing are different in the listed works, the only QoS parameters considered there are bandwidth and number of hops. In our work we add two more QoS parameters, EED and delay jitter.

2.2 Scheduling mechanisms for EED minimization

For both wired and wireless systems, TDMA slot assignment can be considered as a particular case of scheduling in a periodic task system. The problem of EED minimization in periodic task systems has been extensively studied over the past decades for single node and distributed node environments. The results are mostly applied to the systems that run multiple tasks on several processors.

There are two major approaches to perform scheduling of periodic tasks in the distributed systems. One approach is to use rate monotonic (RMS) or priority driven scheduling. A major problem with this approach is that it only guarantees that task can meet its deadline by the end of period, which can make a task that came from the previous node wait almost a whole period before being completed. As a result, the EED between just two nodes may grow to the size of the period.

Another approach is to apply pinwheel scheduling, i.e. to create a schedule whose tasks have not only the

periodic constraints but the distance constraints as well [Huang and Hsueh 2000; Hsueh and Lin 2001; Dong et al 2000a]. The scheduling algorithms with distance constraints were applied by [Dong et al 2000a] to multi-hop TDMA-based system with dynamically arriving message streams. However in that system it was assumed that the schedule of one node does not affect the schedule of another node. Such assumption does not hold for ad hoc environments due to transmission in the air. This pinwheel scheduling minimizes delay jitter between two consecutive task completions. A mechanism called pinwheel phase alignment was used for minimization of the total EED on multiple node scheduling [Huang and Hsueh 2000; Hsueh and Lin 2001]. It was demonstrated [Dong et al 2000b] that the minimization of packet delay jitter results in the minimization of EED.

We apply the distributed EED calculation mechanism proposed in [Dong et al 2000b] to our system in order to provide EED minimization of resulting TDMA schedules. This algorithm is applied with consideration that the slot scheduling on a node may affect the slot scheduling on the other nodes, which is due to the specifics of transmission in the air media.

2.3. The AODV routing protocol

The starting point for our approach is the Ad hoc On-Demand Distance Vector protocol (AODV) [Perkins and Royer 1999]. AODV uses a broadcast route discovery mechanism, and it relies on dynamically established routing table entries at intermediate nodes. The functions performed by the AODV protocol include local connectivity management, route discovery, route table management and path maintenance.

Local connectivity management happens as follows: Nodes learn about their neighbors by either receiving or sending broadcast packets (called "HELLO" packets) from or to their neighbors. Receiving the broadcast or HELLO from a new neighbor or failing to receive HELLO messages from a node that was previously in the neighborhood, indicates that the local connectivity has changed.

The source node initiates path discovery by broadcasting a route request, RREQ, packet to its neighbors. When a node receives RREQ, in case it has routing information, it sends the reply packet, RREP, back to the destination. Otherwise, it rebroadcasts RREQ packet further to its neighbors. As the RREQ packet travels from the source to the destination it automatically sets up the reverse path from all nodes back to the source. As the RREP travels back to the source, each node along the path sets up a forward pointer to the node from which the RREP came and updates its timeout information for route entries to the source and destination.

For each destination of interest a node maintains a single route table entry that contains the address of the destination, the next hop along the path to that destination, the number of hops to the destination, and other route related parameters. If a node is presented with two different routes to the destination it chooses the fresher route. If both routes were discovered simultaneously, the route with fewer hops is preferred. Path maintenance is performed in several ways. When any node lying along the established path moves so that some of the nodes become unreachable, the special RREP message is send to affected source nodes. Upon receiving notification of a broken link, the source node restarts the path discovery process, if it still requires that route.

The design of AODV protocol does not have QoS functionality. Since it is in the class of best-effort ad hoc routing protocols, AODV does not take into account the underlying data link layer and features of transmission over wireless media.

3. QOS-AODV PROTOCOL

We have taken the basic AODV approach and developed an ad hoc QOS protocol called QOS-AODV [Gerasimov and Simon 2002]. While the basic path search algorithm of QOS-AODV is practically the same as one of AODV, QOS-AODV route discovery is augmented with *path bandwidth* allocation and calculation capabilities, allowing QOS-AODV to perform path search with specified bandwidth requirements. The major functions of QOS AODV are connectivity management and link bandwidth information exchange, QOS path discovery and route table management, and VC creation, maintenance and release. Our protocol can perform QoS path discovery for bandwidth constraints only, bandwidth plus EED constraints, and bandwidth plus EED plus jitter-minimization.

3.1. Bandwidth calculation model

This section gives a description of our bandwidth calculation model. Each MAC TDMA frame consists of two phases, a control and data phase. The control phase is divided into number of time slots that equal to the maximum number of nodes in the ad hoc cluster. The latter is defined as a set of mobile hosts that are able to send to and receive from each other. The data phase is used for transmission of application-layer packets and

consists of a limited number of time slots. Each node keeps information about its and its neighbor TDMA slots that are used for send and receive. A node is called a neighbor of another node if those nodes are in the transmission range of each other. We define a node transmission schedule as a sequence of the numbers 0 and 1, where each number corresponds to the order number of the time slot in the data phase. In slot schedule 0 means that slot is free and 1 one means that slot is taken for sending or receiving.

We define node *send slots* and *receive slots* as schedules used by a node for sending and receive, correspondingly. We distinguish between node *free receive slots* and *free send slots*. *Free receive slots* are the slots that are *not* used by a node for sending to and receiving from its neighbors. Since a node should not send at slots that are used by its neighbors for receive in order to prevent the hidden terminal problem, *free send slots* are the slots at which none of a node or its neighbors receive or send.

The *link bandwidth* between two nodes is defined as the intersection of those nodes free slot schedules. The *send link bandwidth* from a node A to a node B is the intersection of node A *free send slots* and node B *free receive slots*. The *receive link bandwidth* at a node A from a node B is, in turn, the intersection of node A's *free receive slots* and node B *free send slots*. We say that there is a *path bandwidth* available from a node A to a node B, if during QoS path discovery session it was possible to allocate a required number of time slots from *link bandwidth* of each pair of nodes along this path AB.

3.2. Protocol description

Connectivity management and exchange of link bandwidth information in QOS-AODV protocol are done using augmented AODV HELLO messages. Each QOS-AODV message contains information about node slots that are used for sending, receive or free for allocation. Once node receives a QOS HELLO message from a neighbor, it recalculates the link bandwidth schedule to all its neighbors, which is the function of nodes' free TDMA slots. HELLO messages are generated either in predefined intervals or shortly after node slot scheduling is changed.

QoS path discovery in QOS-AODV is done the following way. Once an application layer issues a call to create a new Virtual Circuit (VC), a node checks whether it has an entry in the routing table that corresponds to that application call ID. In QoS-AODV route table entries are created on per call ID basis, as opposed to "pure" AODV where a route table entry

corresponds to a destination node ID. Such route table entry to application ID mapping is created since the path bandwidth calculation depends on the number of time slots to be allocated and that number is specific for each VC. If there is no valid routing table entry for call ID then the route request, RREQ, message is broadcasted. Each RREQ message, in addition to AODV information, such as destination ID, number of hops, etc, contains QoS routing data. Those data contain the number of hops to be reserved and path bandwidth parameters that include indexes and link and path bandwidths of up to three hops along the path. Once the node receives RREQ, it calculates a path bandwidth to the source node and if received path bandwidth is sufficient, then it propagates RREQ further with the calculated bandwidth information.

When the destination node receives RREQ and finds out that there is enough bandwidth to the source node, the Virtual Circuit (VC) is created. The destination node initiates reservation message, RSV, that is propagated to the source node using reverse paths from the routing tables corresponding to that source node application call ID. Each node along the path creates the reservation instance that contains the source, destination and application call IDs in addition to the IDs of next and previous node and TDMA slot assignment schedules for send and receive. The information from reservation instances is used for forwarding packets from source to destination node of the VC. Once a node receives the RSV message it tries to reserve time slots from the MAC layer and path bandwidth out of available link bandwidth to the corresponding neighbor. If slot reservation fails then the node generates resource release message USRV that is sent back to the destination. If it succeeds, after the source node receives RSV message and successfully completes required reservations it reports to the application layer, and the latter starts sending of data packets. After all data packets are sent, source node sends URSV message to the destination node and all bandwidth resources are released. See [Gerasimov and Simon 2002] for more details of our protocol.

3.3. EED calculation model

In our EED calculation model, we assume each data packet requires one time slot and the number of packets per frame is the time slot reservation requirement of the call. We present the sketch of our EED calculation model on the Figure 1. The top and bottom lines on the Figure 1 show how the source and the destination nodes application layer packet rates are visualized in TDMA frame.
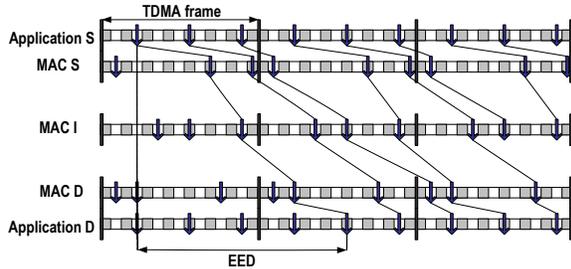
Figure 1. Packet transmission model used for EED calculation. The symbols *S*, *D*, and *I* represent source, intermediate, and destination nodes, correspondingly.

Every node along the VC has a TDMA slot schedule presented as a template of assigned slots in the TDMA frame. As it can be seen on Figure 1, the packet at the source node may wait between the time it is issued by the application layer and the time it is actually sent to the intermediate node *I*. The similar situation happens at the destination node *D*. Once the packet arrives to the node *D* it may wait for some time in order to be accepted by the application layer of that node.

We define EED as the difference between the time a data packet was issued by the application layer of a source node and the time that packet was accepted by the application layer of the destination node. It was shown by [Dong et al 2000b] that the EED is the same for each packet in the system, under the condition that no packets are skipped, at all nodes along the VC. It was proven that every system would reach the state where no packets are skipped in finite amount of time. The EED is the same for every packet in such system because the packets are sent and delivered regularly at the same constant rate at the source and destination nodes respectively.

We define further a *phase* of a given TDMA slot assignment schedule as the distance expressed in time slots between beginning of time frame and the first slot of this schedule. We assume that the phase the source node application layer is constant during the lifetime of VC and it is randomly set for the different VCs.

3.4. QOS-AODV route discovery

Route discovery in QoS-AODV protocol is initiated when a Virtual Connection (VC) is needed for transmission of packets from the application layer of one node to the application layer of another node. During the path discovery session, the QoS-AODV protocol searches for a path that has the required bandwidth and EED. The QOS-AODV performs bandwidth and EED calculations for the entire VC path at every node its path search goes.

The scenario of path discovery in QoS-AODV with EED constraints is as follows: Once the source node receives the QoS call request from its application layer, it needs to establish the VC from the source to the destination nodes in order to transfer application data packets. To do so, the source node checks if it has enough link bandwidth available to any of its neighbors. If it does not, it denies the request for connection at once. Upon success the source node creates a routing table entry for the requested application call ID and the destination address. Finally, the source node broadcasts the AODV request, RREQ, message that in addition to standard AODV information contains call ID, number of slots required for reservation and EED constraint.

When an intermediate node receives a RREQ message it checks whether it already has an entry in the routing table corresponding to received application call ID. If it does not, it creates new route table entry, otherwise it checks whether the received route to the source is fresher then the one it already has. In case it is, it sets information for the reverse path to the source, which corresponds to the given call ID. The route table entry information, in addition to one that corresponds to the original AODV protocol, contains the addresses of three nodes along the path to the source and link and path bandwidth schedules between those nodes. Based on received link and path bandwidth information of nodes along the path, the intermediate node calculates the path bandwidth schedules by the algorithm, which will be described shortly. If the calculated path bandwidth to the source is insufficient, then the node does not forward RREQ message further. Otherwise, it calculates EED and checks whether it satisfies the required value.

When destination node receives an RREQ, it does the path bandwidth and EED calculations the same way the intermediate nodes do. Afterwards, if there is path bandwidth available and EED is satisfactory, it starts the reservation protocol.

For the path bandwidth allocation, we use a heuristic algorithm, which is presented on Figure 2. The path bandwidth allocation algorithm is run on each node after that receives RREQ message. As it is shown on Figure 2, the node calculates the path bandwidth to its *nexthop*, which is immediate hop in the path to the source of RREQ message. For that node uses link bandwidth, *linkBW*, to that hop and two path bandwidths, *pathBW1* and *pathBW2*. *PathBW1* is a path bandwidth from the next hop and the hop, which is next hop's next hop, or *index1*. *PathBW2* is a path bandwidth between *index1* and *index1*'s next hop,

index2. *PathBW1* must be subtracted from link bandwidth *linkBW* in order to prevent collisions of immediate neighbor transmissions. *PathBW2* has to be subtracted from *linkBW* in order to prevent collisions that happen due to "hidden terminal" problem. After the path bandwidths are subtracted, the TDMA slot assignment algorithm, presented in Section 4, is run in order to allocate slots for the path bandwidth to the next hop.

```
INPUT:
sn - number of slots to allocate
nexthop - next hop in the path to the source node
index1 - next hop's next hop in the path to the
source node
index2 - index1's next hop in the path to the source
node
asched - TDMA schedule application layer of a
source node generates packets with
linkBW - link bandwidth to the next hop
pathBW1 - path bandwidth from index1 to the next
hop
pathBW2 - path bandwidth from index2 to index1
OUTPUT:
pathBW –allocated  path bandwidth schedule
{
   // if the source is the next hop then reserve
schedule out of linkBW
   if (source == nexthop) {
       pathBW = reserve_schedule (linkBW, sn,
ashed);
       return pathBW;
   }
   // subtract pathBW1 from linkBW to avoid
collisions
   linkBW = linkBW ^ pathBW1;
   // subtract pathBW2 from linkBW and linkBW1
to avoid "hidden terminal" problem
   linkBW = linkBW ^ pathBW2;
   // do scheduling
   pathBW = reserve_schedule (linkBW, sn,
pathBW1);
   return pathBW;
}
```

Figure 2. Path bandwidth allocation algorithm ran by a node routing layer during route discovery session. The *reserve_schedule* routine calls the TDMA slot scheduling assignment algorithm, which is presented in Section 4.

During the route discovery session, after the node calculates the path bandwidth to the source, it calculates EED using the distributed *Exact-EED-delay algorithm* given in Appendix. As seen on Figure 10,

the EED calculation does not require the knowledge of all TDMA schedules along the VC path. Indeed, only the accumulated EED and the TDMA schedule of a node the EED calculation is being done at is needed. When *Exact-EED-delay* algorithm is run at a node X, it computes a *delay* pair, $<z_X, u_X>$. In this delay pair, z_X is the time, expressed in TDMA slots, between the time that packet was issued by the source node application layer and the time that the packet arrives at node X. u_X, in turn, is the accumulated EED delay at node X, which is also expressed in TDMA time slots. In Figure 3 we show how we apply *Exact-EED-delay* algorithm to our QOS-AODV protocol.

```
INPUT:
asched - TDMA schedule, application layer of a
source node generates packets with
destination - ID of the destination node, to which
EED is calculated
nexthop - next hop in the path to the source node
index1 - next hop's next hop in the path to the
source node
index2 - index1's next hop in the path to the
source node
pathBW - path bandwidth to the next hop
pathBW1 - path bandwidth from index1 to the
next hop
x_X – the phase of TDMA schedule at node X
<z_X, u_X> - the delay pair of node X, where u_X is
EED accumulated at X
OUTPUT:
u_nexthop – EED accumulated for the nexthop node
{
    if (source == nexthop) {
      x_nexthop = get_phase (asched);
      y_nexthop = get_ete (asched, pathBW, x_nexthop);
      < z_nexthop, u_nexthop > = < x_nexthop + y_nexthop,
y_nexthop >;
    } else {
      y_nexthop = get_ete (pathBW1, pathBW,
z_index1);
      < z_nexthop, u_nexthop > = < z_index1 + y_nexthop, u_index1
+ y_nexthop >;
    }
    if (index == destination && pathBW is non-
zero jitter schedule) {
      y_nexthop = get_ete (pathBW, asched, z_nexthop);
      < z_nexthop, u_nexthop > = < z_nexthop + y_nexthop,
u_nexthop + y_nexthop >;
    }
    return u_nexthop;
}
```

Figure 3. EED calculation algorithm used for QoS-AODV protocol. The function *get_ete* calculates the

delay between two consecutive nodes, using algorithm presented on Figure 10 of Appendix. The function *get_phase* returns the schedule *phase*, which is the position of the first slot in that schedule.

As shown on Figure 3, when the EED calculation algorithm is run at a node that is next to the source node, the delay, $y_{nexthop}$, is the number of time slots the first data packet waits at the source node before it is sent to the next node. When the algorithm is run on any other node, $y_{nexthop}$ is the number of time slots between the time the first packet arrived to the *nexthop* of that node, which is identified as z_{index1}, and the time that the packet left that node, which is the sum of $y_{nexthop}$ and z_{index1}. The accumulated EED $u_{nexthop}$ is calculated at a node, which is next to the source, is $y_{nexthop}$, for any other node it is sum of $y_{nexthop}$ and accumulated EED at *index₁* node, u_{index1}. The time which a packet was traveling from the application layer of a source node to the *nexthop* node, or $z_{nexthop}$, is the sum of $y_{nexthop}$ and $x_{nexthop}$ for the node next to source, and $y_{nexthop}$ and z_{index1} for any other node.

If the algorithm is run on the destination node and the arrival schedule has non-zero jitter, then the delay between the time a packet was received by the destination node and the time it was actually accepted by the application has to be calculated and added to the total EED.

Figure 4 presents an example of the QOS-AODV path discovery session, which includes bandwidth and EED calculations.
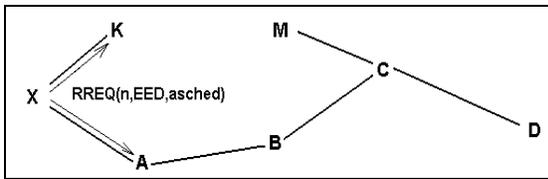

Figure 4a. In order for the source node X to create VC to the destination node D, it broadcasts an RREQ message that contains the number n of slots to allocate and the EED that is required for the VC and X's application layer slot scheduling *asched*.
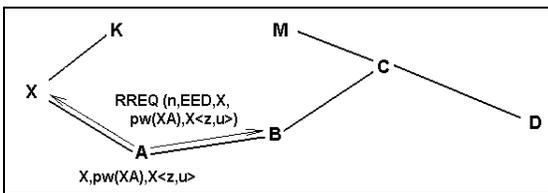

Figure 4b. After A gets RREQ from X, it calculates *path bandwidth* to X, pw(XA), as a part of its receive link bandwidth from X, lw(XA). Using *asched* and pw(XA) node A calculates delay pair $<z_x, u_x>$ where u_x is EED(XA). Then A rebroadcasts RREQ with the address of X, pw(XA) and $<z_x, u_x>$.
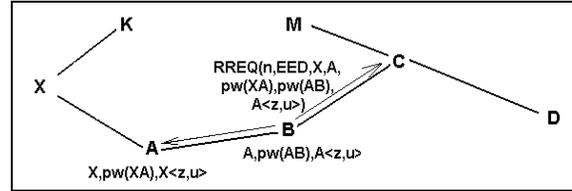

Figure 4c. After B gets RREQ from A and subtracts pw(XA) from lw(AB), it calculates pw(AB) as the part of resulting lw(AB). Using $<z_X, u_X>$ and pw(AB) B calculates delay pair $<z_A, u_A>$ where u_A is EED(XB). Then B rebroadcasts RREQ with the addresses of X and A, path bandwidths pw(XA) and pw(AB), and delay pair $<z_A, u_A>$.

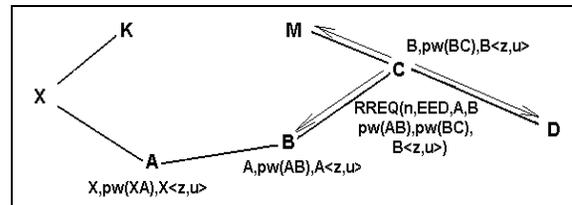**Figure 4d.** When C gets RREQ from B and subtracts pw(XA) and pw(AB) from link bandwidth lw(BC), it calculates path bandwidth pw(BC) as the part of resulting lw(BC). Using delay pair $<z_A, u_A>$ and pw(BC) it calculates delay pair $<z_B, u_B>$ where u_B is EED(XC). Then B rebroadcasts RREQ with the addresses of A and B, path bandwidths pw(AB) and lw(BC), and delay pair $<z_B, u_B>$.

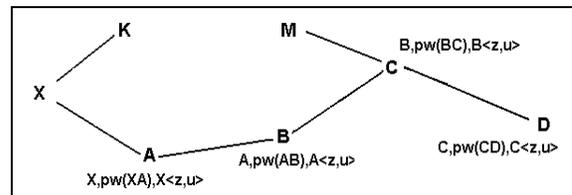**Figure 4e.** When D gets RREQ from C and subtracts pw(AB) and pw(BC) from lw(CD), it calculates path bandwidth pw(CD). Using delay pair $<z_B, u_B>$ and pw(CD) it calculates $<z_C, u_C>$ where u_C is EED(XD). If path bandwidth CD has non-zero jitter, then D calculates another delay pair, $<z_D, u_D>$, using $<z_C, u_C>$ and D's application layer schedule, which is set to *asched* of node X. The EED(XD) is set to u_D in this case.

Figure 4. Example of QoS-AODV with EED constraints path search initiated at source node X for the destination node D. A solid line between two nodes indicates that node can hear each other. The arrow out coming from the node means that a node broadcast messages to its neighbors.

As shown in Figure 4, when a source node X needs to create a VC to destination node D, it broadcasts the RREQ message that contains the number n of slots to allocate, EED that is required for the VC and the application schedule *asched* (Figure 4a). The latter is used by X's application layer to generate the data packets as was described in the Section 3a. When node A gets the RREQ from X, it calculates path bandwidth to X, pw(XA), as a part of its receive link bandwidth from X, lw(XA). In addition, A calculates delay pair $<z_x, u_x>$ using asched and calculated pw(XA), where EED(XA) is u_x. Then A rebroadcasts RREQ with the address of X, pw(XA) and $<z_x, u_x>$ (Figure 4b). When B gets RREQ from A, first, it subtracts pw(XA) from lw(AB) and calculates pw(AB) as the part of resulting lw(AB). Then B calculates delay pair $<z_A, u_A>$ using $<z_X, u_X>$ and pw(AB), where EED(XB) would be u_A. After that B broadcasts RREQ with the addresses of X and A, path bandwidths pw(XA) and pw(AB), and delay pair $<z_A, u_A>$ (Figure 4c). When C gets RREQ from B, it subtracts pw(XA) and pw(AB) from link bandwidth lw(BC) and then it calculates path bandwidth pw(BC) as the part of resulting lw(BC). Using the delay pair $<z_A, u_A>$ and pw(BC) it calculates the delay pair $<z_B, u_B>$, where EED(XC) is u_B. Then B rebroadcasts RREQ with the addresses of A and B, path bandwidths pw(AB) and lw(BC), and delay pair $<z_B, u_B>$ (Figure 2d). When D gets RREQ from C and, it subtracts pw(AB) and pw(BC) from lw(CD), then it calculates path bandwidth pw(CD). Using delay pair $<z_B, u_B>$ and pw(CD) it calculates $<z_C, u_C>$, where EED(XD) is u_C. If path bandwidth schedule pw(CD) has non-zero jitter, then another delay pair, $<z_D, u_D>$, is calculated at D using $<z_C, u_C>$ and D's application layer schedule, which is set to *asched*. The total delay, EED(XD), is set to u_D (Figure 4e).

4. TDMA SLOT SCHEDULING ALGORITHM

Our TDMA slot assignment algorithm was developed in order to minimize the total EED of the VC and to reduce the slot assignment processing. The algorithm is based on two simple ideas. The first one is to find the slot schedule out of available link bandwidth that has zero delay jitter. As it was shown by [Dong et al 2000b], the system with zero delay jitter has lower bound on resulting EED and, as it was shown by [Huang and Hsueh 2000], the phase alignment is possible for the systems with distant constraint scheduling. The second idea is to align the phase of found schedule to the phase of the previous node schedule, so the distance between those phases would be minimal. Our data link TDMA slot reservation algorithm in shown in Figure 5.

```
INPUT:
linkBW – link bandwidth schedule to reserve slots
from
prev_pathBW – path bandwidth to previous node
sn – number of slots to reserve
N – total number of slots in TDMA frame
OUTPUT:
pathBW – reserved path bandwidth
{
   pathBW = 0;
   // generate a schedule with zero jitter
   for (i=1; i<=sn; i++) {
      pathBW |= (1 << (N/sn)*i);
   }
   // calculate the phase of schedule to the previous
node
   first_slot_number = get_phase (prev_pathBW);

   // shift path bandwidth schedule, so its first slot
is positioned  right after the first slot of schedule
to previous node
   pathBW <<= (N/sn – first_slot_number - 1);

   // starting from the shifted position check whether
such a schedule/ is available in the given link
bandwidth, and if it is not,  then keep shifting
schedule to the right
   for (i=0; i<(N/sn – first_slot_number - 1); i++) {
      if (slots_number(linkBW & pathBW) == sn)
         return pathBW;
      pathBW >>= 1;
   }
   // if the schedule is not found yet, then shift it, so
the first slot would be in the most left position.
Keep shifting the schedule until available schedule
is found or the first slot reaches the position of the
first slot of bandwidth to the previous node
   pathBW <<= (N/sn – 1);
   for (i=0; i < (first_slot_number + 1); i++) {
      if (slots_number(linkBW & pathBW) == sn)
         return pathBW;
      pathBW >>= 1;
   }
   return 0;
}
```

Figure 5: Zero Jitter with Phase Alignment TDMA slot scheduling algorithm

As shown in Figure 5, the ZJPA algorithm initially generates the TDMA slot scheduling template, *pathBW*, which has zero jitter, i.e. the distance between slots is equal to the total slots number divided by number of slots to reserve. Then the phase of the previous node path bandwidth schedule, or *prev_pathBW*, is calculated as the distance between beginning of TDMA frame and the first slot in that schedule. After that, the path bandwidth scheduling template, *pathBW*, is shifted, so its first slot is positioned right after the first slot of *prev_pathBW*. We check then if in the link bandwidth, *linkBW*, all slots from schedule *pathBW* are available. If at least one slot that contained in *pathBW* is busy in *linkBW*, we shift *pathBW* template to the right and check again. We keep shifting *pathBW* until available schedule is found or initial schedule, which we start from before the shift, is received. If available schedule can not be found, then we shift the template, so the first slot would be in the most left position of frame and keep shifting the *pathBW* to the right until available schedule is found or the first slot of *pathBW* reaches the position of the first slot of *prev_pathBW*.

Figure 6 presents an example of finding path bandwidth using link bandwidth with the consideration of previous path bandwidth.
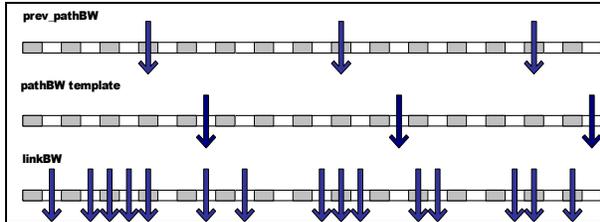

Figure 6a. Path bandwidth template, *pathBW*, which has a zero delay jitter schedule composed.
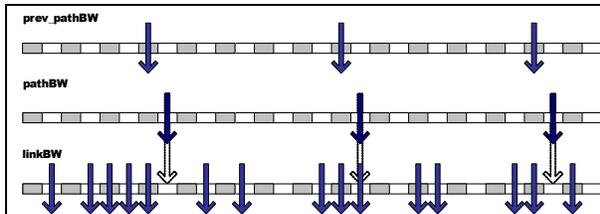

Figure 6b. *PathBW* is aligned with *prev_pathBW*, so that EED between these schedules is equal to one TDMA time slot.
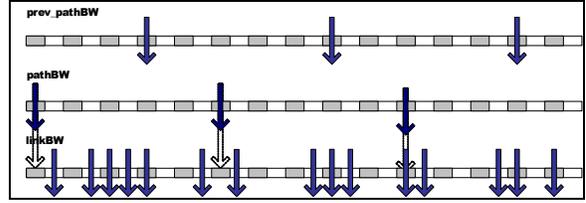

Figure 6c. After *pathBW* was shifted to the right and no available schedule was found it is shifted to the most left and then it is kept shifted to the right again.
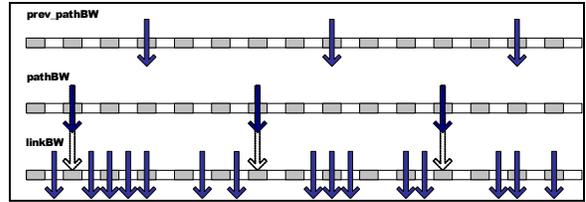

Figure 6d. The available slots are found. This schedule produces EED equal to 6 TDMA time slots.

Figure 6. Example of how path bandwidth is calculated when employing ZJPA algorithm. The TDMA frames with 30 slots shown for previous path bandwidth, or *prev_pathBW*, link bandwidth, or *linkBW*, and path bandwidth template, or *pathBW*. Solid arrows represent slots that are allocated in TDMA frames.

The task represented on Figure 6 is to allocate three TDMA frame slots out of available link bandwidth, *linkBW*, so that a new path bandwidth schedule would have zero delay jitter and minimum available EED. First, the path bandwidth template, *pathBW*, which has a zero delay jitter schedule composed (Figure 6a). Then the phase of *pathBW* is aligned with linkBW schedule so that the delay between those two schedules would be 1 TDMA time slot (Figure 6b). After *pathBW* was shifted to the right and no available schedule was found, it is shifted to the most left and then it is kept shifted to the right again (Figure 6c). Finally, the available schedule is found (Figure 6d), and it produces EED equal to 6 TDMA time slots.

5. PERFORMANCE EVALUATION

First, we have evaluated the performance of QOS-AODV protocol, which does not have EED constraints. For this, we performed series of simulations on ns-2 [Ns-2] for this protocol and its two simpler derivatives. We used the CMU extension of ns-2 simulator that has already implemented the AODV protocol. Since the existing implementation of AODV relies on the underlying MAC 802.11 and there was no suitable implementation of TDMA MAC

in ns-2 we implemented a new MAC TDMA layer. We developed an interface between MAC and the routing layers that enables slot reservation for sending to and receiving from the specified neighbors. Further, we refer to QOS-AODV protocol with EED constraints as *QOSA-AODV* for simplicity.

The two protocols that we compared to *QOSA-AODV* are as follows. The first protocol is called *PURE-AODV*. Its functionality is close to one in the original AODV, with only exception that VC is created for sending packets from the source to the destination. As the original AODV, *PURE-AODV* does not maintain the information about link bandwidth available to its neighbors. After the source node gets a route to the destination it starts reservation protocol. From the routing table entry corresponding to that route it gets the address of the immediate neighbor then guesses path bandwidth to it. If the guess is right, it makes reservations, as described for *QOSA-AODV*, and then sends RSV message to the next hop along the path. When intermediate hop receives RSV message, it guesses the path bandwidth scheduling to the next hop, from the full link bandwidth range with the exclusion of the path bandwidth received from the source and if that guess is right, it propagates the RSV message further. In case the guess was wrong, the URSV message is composed and sent back to the source node. When RSV message reaches the destination and all the reservations went successfully, the rest proceeds as in *QOSA-AODV*. The resource release and handling of broken routes happen the same way as in *QOSA-AODV*.

The second protocol is called Simple QoS AODV, or *QOSS-AODV*. The path discovery and maintenance in this protocol, together with the VC creation and reservation mechanisms, happen as in *PURE-AODV*. The difference is that in *QOSS-AODV*, the nodes are aware of the available link bandwidth to their neighbors, and exchange link bandwidth information as in *QOSA-AODV*. While doing reservations, *QOSS-AODV* composes a path bandwidth scheduling out of known link bandwidth to the next hop. The above-described properties of the protocols are summarized in the Table 1.

In order to evaluate the route discovery capabilities of *QOSA-AODV* with EED constraints, we compared it with *QOSS-AODV*. The functionality of the latter protocol is the same as described above with only exclusion that during the reservation phase it would not create a VC if the EED parameter, in addition to bandwidth, does not satisfy defined QoS requirements for that VC.

Protocol Characteristics	PURE-AODV	QOSS-AODV	QOSA-AODV
Time slot information exchange	NO	Yes, in AODV HELLO messages	
Path discovery	RREQ, RREP as in original AODV		RREQ contains additional QOS information
Route table management	One route table entry per destination as in original AODV		One route table entry per call ID
Reservation protocol	RSV is sent from source to destination, then ACK is sent from destination to source		RSV is sent from destination to source

Table 1. Properties of *PURE-AODV*, *QOSS-AODV* and *QOSA-AODV* protocols

In order to evaluate the performance of our Zero Jitter with Phase Assignment, ZJPA, algorithm we have implemented three other slot reservation and scheduling algorithms: Random, FIFO and Zero Jitter-FIFO TDMA slot assignment algorithms.

The first algorithm, called Random TDMA slot assignment, takes a link bandwidth and randomly allocates the number of required slots to the resulting path bandwidth, as it is shown on Figure 7.

```
INPUT:
sn – number of slots to reserve
linkBW –given  link bandwidth schedule
SN – number of slots in TDMA frame
OUTPUT:
pathBW – reserved path bandwidth
{
    pathBW = 0;
    while (sn) {
        // get a random slot, then check whether it is
reserved already. Reserve this slot, if it is free
        mask = 1 << Random::integer(SN);
        if (mask & (pathBW | linkBW))
            continue;
        pathBW |= mask;
        count--;
    }
    return pathBW;
}
```

Figure 7. Random TDMA slot assignment algorithm

The second algorithm, called FIFO, is presented on Figure 8. This algorithm makes reservations of first available slots from the given link bandwidth.

```
INPUT:
sn - number of slots to reserve
linkBW – given link bandwidth schedule
SN – number of slots in TDMA frame
OUTPUT:
pathBW - reserved path bandwidth
{
    pathBW = 0;
    // create a schedule template that has 0 jitter
    for (i=1; i<=sn; i++) {
        pathBW |= (1 << (SN/sn)*i);
    }
    // align the schedule so the first slot is
positioned at the beginning of the frame. Moving
to the left, check whether link bandwidth has
available slots at the required positions. Reserve
the first available schedule.
    pathBW <<= (SN/sn - 1);
    for (i=0; i < (SN/sn - 1); i++) {
      if (slots_number(linkBW & pathBW) == sn)
          return pathBW;
      pathBW >>= 1;
    }
    return 0;
}
```

**Figure 8.** FIFO TDMA slot assignment algorithm

```
INPUT:
sn - number of slots to reserve
linkBW -given  link bandwidth schedule
SN - number of slots in TDMA frame
OUTPUT:
pathBW - reserved path bandwidth
{
   pathBW = 0;
  //starting from the most left slot in the TDMA
  frame, check whether it is free in the link
  bandwidth. Reserve the slot if it is available.
  mask = 1 << SN;
  for (i = SN-1; i>=0; i--) {
     mask >>= 1;
     if (!(linkBW & mask))
        continue;
     pathBW |= mask;
     if (!-sn)
        break;
  }
  return pathBW;
}
```

**Figure 9.** Zero Jitter-FIFO TDMA slot assignment algorithm

The third algorithm, called Zero Jitter – FIFO, is presented on Figure 9. In this algorithm first available slots that confirm to zero jitter slot allocation requirement are reserved.

The performance of three protocols was studied for different system topology and mobility models. The mobility pattern was set as following: in the beginning of the simulation the nodes randomly distributed on the specified area. Then for each node, the destination is randomly picked, and the node starts moving towards that destination with the speed not exceeding a predetermined one. When node stops it randomly selects a new destination and further procedures are performed the same way as before. For the call generation the destination and the source nodes were randomly picked and the time for ith call to be issued was calculated as $stime(i) = stime(i-1) + interval*0.75 + random\_uniform(0, interval*0.5)$, where $stime(0) = 0$, and $interval = total\ simulation\ time/total\ number\ of\ calls$, and $random\_uniform()$ returns a random number from the interval specified in its parenthesis. This was done to ensure that calls do not arrive simultaneously.

The topology of the system was created by the random generation of sets of nodes, varying from 10 to 30, for the different square areas varying from 500x500sq.m (10 nodes) to 700x700sq.m (20 nodes) to 900x900sq.m (30 nodes). The radius of node transmission was set to be equal to 250m. The simulation time was set to 40,000 seconds and the number of calls was set to 10,000 per simulation. The holding time of the connection request followed the normal distribution with the average of 20 second with the standard deviation of 10 seconds. The number of TDMA time frames required per connection was randomly set to 1, 2, or 3 slots. The node speed was varied from 0 to 30m/s. The results are presented on Plots 1-3.
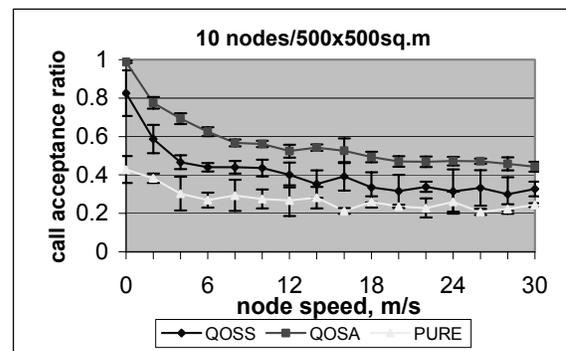
Plot 1. Call acceptance ratio for different node speeds for 10 nodes placed on 500x500 meter area.
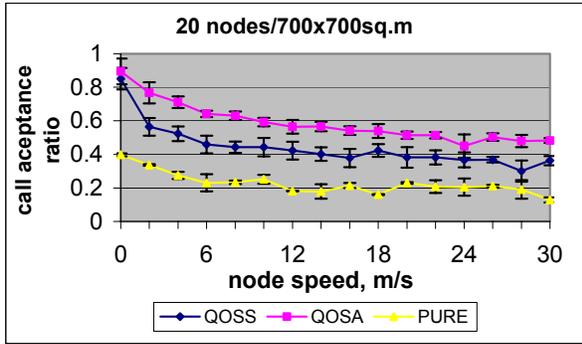
Plot 2. Call acceptance ratio for different node speeds for 20 nodes placed on 700x700 meter area.
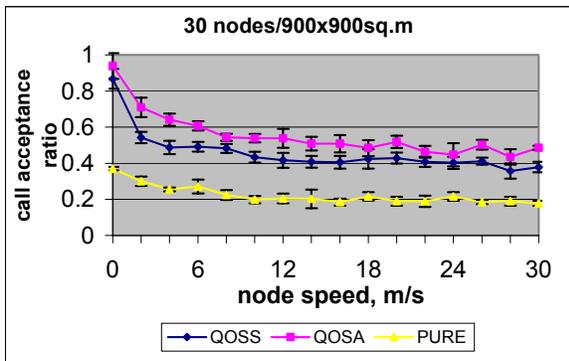
Plot 3. Call acceptance ratio for different node speeds for 30 nodes placed on 900x900 meter area**.**

Each point shown on the presented plots is the average of data taken from five simulation experiments; the error bars are the standard deviation of those data. As it can be seen, for the stable nodes, most of the calls are accepted for *QOSA-AODV*, while *QOSS-AODV* and *PURE-AODV* acceptance rates are 85% and 40%, correspondingly. As the nodes move, the acceptance rate for all protocols decreases because of increasing number of race conditions, which may occur during the VC establishment process. The race condition may happen during the reservation phase, and is due to outdated bandwidth scheduling. This situation arises because other nodes have reserved the same slots, or the neighborhood of nodes along path has changed due to node movements that as the result change node link bandwidth.

The results show that the call acceptance ratio for *PURE-AODV* is the lowest and is around 20% for all three topology setups. It also can be seen that *QOSA-AODV* outperforms *QOSS-AODV* for at least 10% for all topologies presented.

One advantage of *PURE-AODV* is that it has the least control message overhead during its local connectivity management and path discovery sessions. *PURE-AODV* starts reservations for every call it receives from the application layer. However, unawareness of *PURE-AODV* about link bandwidths to the node neighbors may quickly lead to race conditions.

The *QOSS-AODV* connectivity management makes nodes aware of the link bandwidth to their neighbors. However, nodes do not know whether there is a path bandwidth available to the required destinations. As in *PURE-AODV* the reservation process starts for most of the call as soon as there is enough link bandwidth available between node and the next hop of corresponding route table entry. During the reservation process, node may soon discover that while there is a path to the destination there is no path bandwidth available, so the reservation backs up to the source and the call fails.

In *QOSA-AODV* once the path is discovered it is guaranteed that the reservation done along that path would not be refused because of insufficient bandwidth. So, all the reservations started are completed except ones that encounter racing conditions. Despite the message overhead of QOSA, the significant increase in call acceptance makes it clearly superior to the other protocols we studied.

For the evaluation of *QOSA-AODV* with EED constraints and proposed TDMA ZJPA algorithm, we compared the performance of this protocol with that of *QOSS-AODV*. We created the system of 20 nodes placed on 700x700sq.m area. The radius of node transmission was 250m. Each simulation lasted for 40,000 seconds and the number of calls was set to 10,000 per simulation. The holding time of the connection request followed the normal distribution with the average of 20 second with the standard deviation of 10 seconds. The number of TDMA time frames required per connection was randomly set to 1, 2, or 3 slots. The node speed was varied from 0 to 30m/s. The total EED requirement per call was randomly generated and equal to ½, 1, 2 TDMA frames, and unlimited EED requirement.

The performance of *QOSA-AODV* and *QOSS-AODV* protocols together with slot assignment algorithms were studied as dependency of call acceptance ratio for the different EED requirements. Some of the results are presented on Plots 4-7.

CAR for QOSS (no EED)
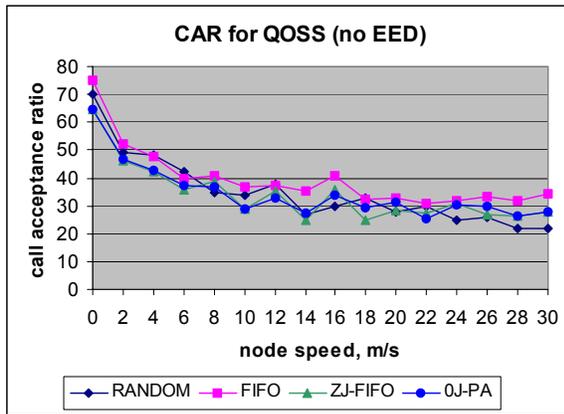
Plot 4. Dependency of the Call Acceptance Ratio (CAR) from the node speed for the *QOSS-AODV* system with is no EED requirement for Virtual Circuits.
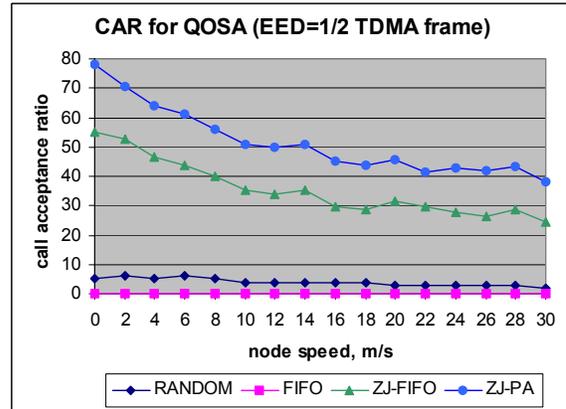
CAR for QOSA (no EED)
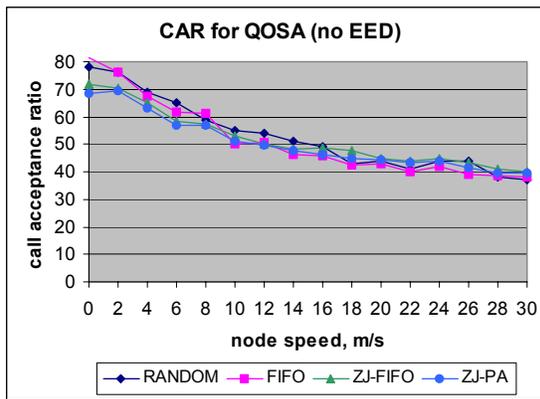
Plot 5. Dependency of the Call Acceptance Ratio (CAR) from the node speed for the *QOSA-AODV* system with is no EED requirement for Virtual Circuits.

CAR for QOSS (EED=1/2 TDMA frame)
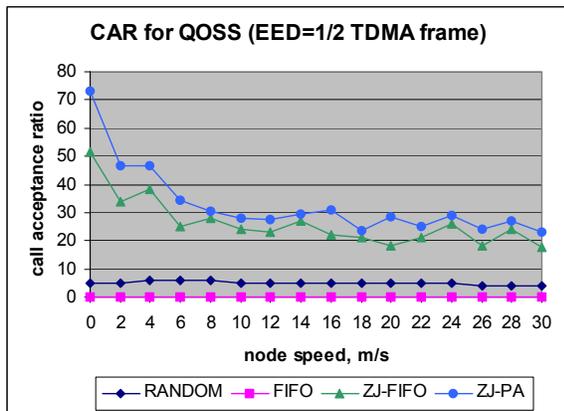
Plot 6. Dependency of the Call Acceptance Ratio (CAR) from the node speed for the *QOSS-AODV*

system with EED requirement equal to ½ of TDMA frame

CAR for QOSA (EED=1/2 TDMA frame)

Plot 7. Dependency of the Call Acceptance Ratio (CAR) from the node speed for the *QOSA-AODV* system with EED requirement equal to ½ of TDMA frame

As it is seen on Plots 4 and 5, the call acceptance ratio for *QOSA-AODV* protocol is significantly higher then the call acceptance ratio for *QOSS-AODV* protocol and the difference is in the range of 10-20%. In our previous work, the TDMA algorithm, used for the slot assignment was simple FIFO. As it is seen from these plots, all shown slot assignment algorithms produce about the same call acceptance ratio within both *QOSA-AODV* and *QOSS-AODV* protocols.

Plots 6 and 7 present the Call Acceptance Ratio for the system with stricter total EED requirement that is equal to ½ of TDMA frame, which means that the call is rejected if it is total EED between source and destination nodes is bigger then ½ of TDMA frame. The graphs on Plot 7 show that *QOSA-AODV* algorithm with ZJPA slot assignment produce the best acceptance ratio, which is actually equal to one that is produced by the system with no EED requirements. Plots 6 and 7 also show that the call acceptance ratio produced by Random and FIFO slot assignment algorithms for both *QOSS-AODV* and *QOSA-AODV* are quite low and is equal to 0%(FIFO) and 5%(Random), correspondingly. Such serious deviation of CAR produced by FIFO and can be explained by the fact that during slot allocation at each node first available slots of TDMA frame are allocated, which simply leads to EED close to one TDMA frame per hop.

In addition, we measured the average total and per hop EED for both *QOSS-AODV* and *QOSA-AODV*. We define average total EED as the sum of all source-to-

destination EED components divided by the number of calls. We define per hop average EED as the average of total EED of the call divided by the number of hops of that call VC. Both EED metrics were taken at the time the calls were accepted. Since these metrics are calculated from TDMA schedules that are created during path discovery session, they do not depend from the node speed. We present received EED in Tables 2 and 3.

Slot Assignment	AODV Type	EED requirement			
		no EED	2 frames	1 frame	1/2 frame
Random	QOSS	38	35	26	13
	QOSA	39	36	26	13
FIFO	QOSS	36	35	29	n/a
	QOSA	36	35	29	n/a
0J-FIFO	QOSS	17	16	13	8
	QOSA	14	14	11	8
0J-PA	QOSS	5	5	5	5
	QOSA	3	3	3	4

Table 2. Average total EED received for different EED requirement requirements, which applied for the systems with different TDMA slot assignment schedules.

Slot Assignment	AODV Type	EED requirement			
		no EED	2 frames	1 frame	1/2 frame
Random	QOSS	21	21	20	13
	QOSA	20	20	20	13
FIFO	QOSS	21	21	21	n/a
	QOSA	20	20	20	n/a
0J-FIFO	QOSS	9	9	8	5
	QOSA	7	7	7	5
0J-PA	QOSS	3	3	3	2
	QOSA	1	1	1	1

Table 3. Average per hop EED received for different EED requirement requirements, which applied for the systems with different TDMA slot assignment schedules.

As it is seen on Tables 3 and 4, FIFO and Random slot assignment algorithms provide the largest total and normalized EED for both *QOSS-AODV* and *QOSA-AODV* protocols. ZJPA algorithm produces the lowest total and normalized EED, which in case of *QOSA-AODV* protocol is lower then for *QOSS-AODV* protocol.

6. SUMMARY

In this work, we developed a protocol that supports the QoS requirements of tightly-coupled distributed applications in a mobile ad hoc environment. Our targeted system is on the order of 10 to 30 nodes, and our targeted connection lifetimes are on the order to a few minutes. We expect that, in practice, tightly

constrained ad hoc applications will need to continually "open" and "close" new connections, in order to deal with route breaks in an ad hoc environment. We then evaluated the performance of this protocol through simulation and comparison with simpler QOS version of AODV which path discovery session is similar to one of "pure" AODV. We demonstrated that QOSA call acceptance ratio is about 20% higher then call acceptance ratio of QOSS.

In addition, we presented a simple heuristic TDMA slot assignment algorithm that significantly reduces total EED of VC calls. As our simulations have shown, the call acceptance ratio of this algorithm is about the same as one for basic call acceptance algorithm with the system with no EED requirements. We demonstrated that in the systems with tougher EED requirements, the call acceptance ratio produced 0JPA algorithm is significantly higher then one in the systems with basic TDMA slot assignment algorithms. We believe that our approach can satisfy tight synchronization requirements of tightly coupled distributed applications.

REFERENCES

Bevilacqua A. 1999, "A Dynamic Load Balancing Method On A Heterogeneous Cluster Of Workstations," Informatica, Vol. 23, number 1 1999.

Cansever D.H., Michelson A.M., Levesque A.H. 1999, "Quality of service support in mobile ad hoc IP networks", GTE Labs. Inc., Waltham, MA, USA. IEEE Military Communications Conference Proceedings, 1999. MILCOM 1999, page(s): 30 - 34 vol.1 31 Oct.-3 Nov. 1999.

Chen S. and Nahrstedt K. 1999, "Distributed quality-of-service routing in ad hoc networks", IEEE Journal on Selected Areas in Communications, page(s): 1488 - 1505 Aug. 1999

Dong L., Melhem R., Mosse, D. 2000a, "Scheduling algorithms for dynamic message streams with distance constraints in TDMA protocol"; 12th Euromicro Conference on Real-Time Systems, 2000. Euromicro RTS 2000, pages 239 -246

Dong L., Melhem R., Mosse D. 2000b, "Effect of scheduling jitter on end-to-end delay in TDMA protocols"; Seventh International Conference on Real-Time Computing Systems and Applications, 2000. Proceedings, 2000, Page(s): 223 -230

Gerasimov I. and Simon R. 2002, "A Bandwidth-Reservation Mechanism for On-Demand Ad Hoc Path Finding". IEEE-SCS 35th Annual Simulation Symposium. San Diego, CA, April 2002, pages 27-34.

Ho Y.-K. and Liu R.-S. 2000, "On-demand QoS-based routing protocol for ad hoc mobile wireless networks". Fifth IEEE Symposium on Computers and Communications, 2000. Proceedings. ISCC 2000, pages: 560 - 565 3-6 July 2000

Hsueh C.-w. and Lin K.-J. 2001; "Scheduling Real-Time Systems with End-to End Timing Constraints Using the Distributed Pinwheel Model"; IEEE Transactions on Computers, vol. 50, 2001; pages: 51-66

Huang Y.-S. and Hsueh C.-w. 2000; "Minimizing the maximum End-to-End Delay on Tree Structure Using the Distributed Pinwheel Model"; Seventh International Conference on Real-Time Computing Systems and Applications, 2000. Proceedings. 2000, pages: 127-134

Lin C. R. and Liu J.-S. 1999, "QoS routing in ad hoc wireless networks", IEEE Journal on Selected Areas in Communications, page(s): 1426 - 1438, Aug. 1999, Volume: 17 Issue: 8

Lin, C.R. and Liu, C.-C. 2000, "An on-demand QoS routing protocol for mobile ad hoc networks", Conference on IEEE International Networks, 2000. (ICON 2000) Proceedings, pages: 160 - 164, 5-8 Sept. 2000

Lumetta S., Mainwaring A., Culler D. 1997, "Multi-Protocol Active Messages on a Cluster of SMPs", Steve Lumetta, Alan Mainwaring and David Culler, Supercomputing97, Nov. 1997

Ns2; see http://www.isi.edu/nsnam/ns/.

Perkins C.E. and Royer, E.M. 1999; "Ad hoc on-demand distance vector routing", Second IEEE Workshop on Mobile Computing and Applications, 1999, Proceedings. WMCSA '99, pages 90-100.

Stallings W. 2002, *Wireless Communications and Networks,* Prentice Hall, 2002

Zheng H., Buyya R., Bhattacharya S. 1999, "Mobile Cluster Computing and Timeliness Issues", Informatica: An International Journal of Computing and Informatics, Vol. 23, No. 1, 1999.

Xiao H., Seah W.K.G., Lo A., Chua, K.C. 2000, "A flexible quality of service model for mobile ad hoc networks", IEEE 51st Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring, Tokyo. 2000, pages: 445 - 449 vol.1 15-18 May 2000.

APPENDIX

In the Appendix, we briefly summarize *Exact-EED-delay* and *Construct-pairs* algorithms, whose detailed description is given in [Dong et al 2000b].

The notations used in presented algorithms are as following. The packets are transferred as message streams, each of which requires n time slots per TDMA frame. Set of slots allocated at a node N_k for a message stream is defined as $S = \{S_1^k, S_2^k, ..., S_n^k\}$. A local delay pair at node N_k is (S_j^{k-1}, y_j), where $y_j = S_i^k - S_j^{k-1}$ is a difference in time slots between times a packet was received and sent at node N_k. All delay pairs related to a single message stream at a node N_k comprise a delay pair set, or $P_k = \{(S_j^{k-1}, y_j), j \in [1, n]\}$.

1. At the source node N_1, assume that a local delay pair (x_1, y_1) is in P_1, indicating that an input at x is delayed at N_1 by y slots and is transmitted at time z=x+y. Form an accumulated delay pair <z,y>.
2. Transmit accumulated delay pair to the next node.
3. When node N_k (1<k<=h) receives the accumulated delay pair <z,u>, it finds a local delay pair (x_j, y_j) in P_k such that $x_j=\text{mod}(z)$. The accumulated delay pair is updated to <z+y_j, u+y_j> and transmitted to the next node.
4. When the destination node receives <z,u> and executes the previous step the resulting delay pair <z,u> provides EED as u.

Figure 10. Exact-EED-delay algorithm

1. Assume that n packets arrive at N_k in the slots $S = \{S_1^{k-1}, S_2^{k-1}, ..., S_n^{k-1}\}$. Find $\{z_1, ..., z_n\}$, the set of time slots for N_k to transmit these n packets if each packet is transmitted immediately in the next available allocated slot at N_k.
2. Assume that $z_n = S_g^k$, which means that the nth received packet is transmitted by N_k at the location of the gth instance. Then $\{S_{g-n+1}^k, ..., S_g^k\}$ is the set of time slots at which Nk will transmit each packet without slot skipping. N_k will transmit the jth packet received at time S_j^{k-1} at time S_{g-n+j}^k, where j ∈[1,n].
3. Construct the set P_k with pairs (S_j^{k-1}, y_j) for all j ∈ [1,n], where $y_j = S_{g-n+j}^k - S_j^{k-1}$

Figure 11. Construct-pairs algorithm

Irina Gerasimov

Irina Gerasimov received her BS degree in Chemistry from Moscow State University, in 1996, and MA degree in Physical Chemistry from the Johns Hopkins University in 1998. She is currently a PhD student at the Computer Science Department of George Mason University in Fairfax, VA, USA. Her research interests are in mobile and wireless systems, ad hoc networks and mass storage systems. She has published several technical papers in these areas. She is also the co-founder of Data Foundation Inc. and TechnoMages Inc., Maryland-based companies specializing in development and manufacturing of Data Storage and Storage Area Networking sub-systems.

Robert Simon

Dr. Robert Simon received a B.A. in History and Political Science from the University of Rochester, and a Ph.D. in Computer Science from the University of Pittsburgh in 1996. He is currently an Associate Professor in the Department of Computer Science at George Mason University in Fairfax, VA, USA. His research interests are in networks, real-time and distributed systems. He has published over 50 peer-reviewed conference and journal papers in these areas. He has served on numerous program committees and review panels, and was the Program Chair for the SCS Communication Networks and Distributed Systems conference in 1999, 2000 and 2001.