# Design and Experimental Evaluation of a Cross-Layer Deadline-Based Joint Routing and Spectrum Allocation Algorithm

Jithin Jagannath, Sean Furman, Tommaso Melodia, and Andrew Drozd, *Fellow, IEEE*

**Abstract**—The design and implementation of a novel distributed deadline-based routing and spectrum allocation algorithm for tactical ad-hoc networks is reported in this article. Different traffic classes including text, voice, surveillance video, and threat alert among others need to be handled by these networks. Each of these traffic classes have different quality of service (QoS) based deadline requirements. Additionally, these networks are characterized by dynamic channel and traffic conditions that vary with time and location. Even under these conditions, it is critical to receive packets before the deadline expires to make rapid decisions in the battlefield. Therefore, a tactical ad-hoc network should be able to adapt to these requirements and maximize the number of packets delivered to the destination within the specified deadline. A distributed deadline-based routing and spectrum allocation algorithm is designed to maximize the utilization of the available resources and ensure delivery of packets within the deadline constraints. To this end, a weighted virtual queue (VQ) that is used to construct the network utility function is defined. Accordingly, the optimal session, next hop, transmit power, and frequency is determined by the distributed algorithm to ensure efficient utilization of the available resources. Hence, maximizing the delivery of packets to the intended destination within the specified deadline. The 49 node simulation shows up to 35 percent improvement in effective throughput and 26 percent improvement in reliability as compared to joint ROuting and Spectrum Allocation algorithm (ROSA), which does not adapt according to the deadline requirements of the data flowing through the network. As a secondary objective, this work advances the state of the art of the experimental cross-layer framework to address the challenges involved in having such cross-layer algorithms implemented on a testbed. The required flexibility to change the transmission parameters on-the-fly is provided by the proposed framework. The network is designed to enable the data exchange between neighbors using custom designed control packets (which might be different for different algorithms) since this information is critical for nodes to perform optimization. Cross-layer optimization is achieved by means of data management and control entities that enable information exchange between layers. The practicality of the proposed solution was proven by having the novel algorithm implemented on a five-node software defined radio testbed which leverages the proposed cross-layer framework. In contrast to ROSA, the proposed algorithm demonstrated up to 17 percent improvement in terms of throughput and reliability. The performance improvement achieved is expected to increase on a larger network deployment.

**Index Terms**—Deadline-based routing, cross-layer optimization, cognitive radio, resource allocation, software defined radio, USRP testbed

## 1 Introduction

In a tactical ad-hoc network, there exists a constant tension between available resources and the required quality of service (QoS) performance. Nodes in the network have to deal with severe interference, spectrum crunch, adversarial jamming and changing network topologies. Additionally, a typical tactical network as depicted in Fig. 1 is required to handle various traffic classes including regular sampling data, voice, surveillance video, threat alert, among others. Each of these traffic classes have substantially different QoS based deadline requirements. For example, periodic surveillance data might have looser deadline constraints when compared to a video or threat alert message. In these delay-intolerant networks, only packets that arrive at the destination within the specified deadline are viable and contribute to the overall network throughput. In these scenarios, it becomes important to examine the interaction between spectrum management, routing and session management to develop cross-layer control algorithms capable of maximizing the effective throughput of the network. In this paper, we consider only the packets that arrive at the destination within the specified deadline in the computation of *effective throughput*.

Cognitive radio technology along with various dynamic spectrum access (DSA) techniques have been proposed to improve the spectrum utilization of the network by enabling opportunistic access of free spectrum chunks. In previous work [2], an optimization algorithm (ROSA) was proposed to jointly select route and spectrum such that overall network throughput is maximized. This algorithm combines the idea of backpressure algorithm [3] with channel dependent

- *J. Jagannath is with ANDRO Advanced Applied Technology, ANDRO Computational Solutions, LLC, Rome, NY 13440, and the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115. E-mail: jjagannath@androcs.com.*
- *S. Furman and A. Drozd are with ANDRO Advanced Applied Technology, ANDRO Computational Solutions, LLC, Rome, NY 13440. E-mail: {sfurman, adrozd}@androcs.com.*
- *T. Melodia is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115. E-mail: melodia@ece.neu.edu.*

*Manuscript received 19 Dec. 2017; revised 16 May 2018; accepted 14 Aug. 2018. Date of publication 20 Aug. 2018; date of current version 28 June 2019. (Corresponding author: Jithin Jagannath.)*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TMC.2018.2866093*

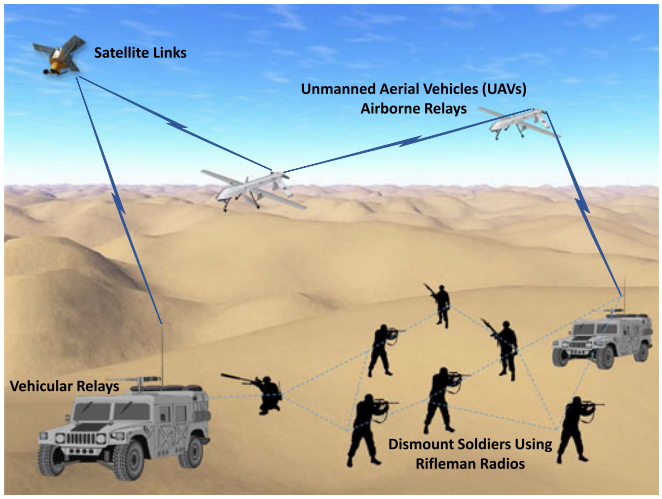


Fig. 1. Tactical ad-hoc network.

opportunistic routing. Simulations show that ROSA outperforms the traditional algorithms that use either dynamic spectrum allocation with fixed route or dynamic routing with fixed spectrum allocation. In this work, we substantially extend [2] to examine the network performance in terms of effective throughput and reliability for multiple sessions with different deadline constraints. This enables such routing and resource allocation algorithms to handle various QoS based traffic classes efficiently. Accordingly, we develop a distributed deadline-based optimization algorithm for tactical ad-hoc networks. Some of the challenges in designing a deadline-based algorithm are as follows:

- Each node has to carefully manage multiple sessions to meet the deadline requirements. For example, sessions with longer backlogs and larger deadlines can be held back while accelerating shorter backlogged-smaller deadline sessions.
- Adopting an effective resource allocation procedure that would negotiate the access of medium and choose optimal transmission parameters. In a large network, the spectrum occupancy varies based on location and time, thus nodes may have to use different parts of the spectrum in order to route a session in the most effective manner.
- Choosing appropriate routes to meet the needs of each session belonging to different traffic classes.
- The network should be able to adapt to broken routes or failed nodes by choosing alternate paths.
- The design should be scalable, reduce communication overhead and yet enable the network to adjust dynamically to the available resources. Therefore, it is critical to design a distributed approach that is feasible on a practical network.

Therefore, the overall objective of this work is to design and evaluate a distributed algorithm that utilizes the available resources to determine optimal route, session and spectrum to deliver maximum number of packets to their intended destination within the specified deadline. The weighted virtual queue (VQ) used in cross-layer optimization ensures proper management of the sessions. The virtual queue length (VQL) takes into account deadlines associated with each packet. The joint routing and spectrum allocation aspects of the algorithm provides optimal resource allocation and enables opportunistic routing. The distributed nature of the proposed algorithm along with forward progress based routing helps the network to recover from broken routes or failed nodes. These features are critical in any delay-intolerant applications and will be especially useful in tactical ad-hoc networks where the delayed delivery of critical information in a multihop network can be fatal.

## 2 Related Work

Dynamic spectrum allocation has been widely investigated with the objective to maximize spectrum utilization and is mainly divided into centralized [4], [5] and distributed [2], [6] approaches. While spectrum allocation techniques are designed to improve spectrum utility based QoS [7], [8], [9], queue length based backpressure (Q-BP) scheduling algorithm was first proposed in [3] and was shown to be throughput optimal in terms of achieving network stability under any feasible load. It is well known that the Q-BP algorithm suffers from high computational complexity and the last packet problem. To reduce the computational delay for practical implementation, a greedy maximum scheduling (GMS) algorithm is studied in [10], [11], [12]. This algorithm first chooses the link $l$ with maximum weight from the set of all links $\mathcal{S}$ and eliminates links that interfere with $l$ from the set $\mathcal{S}$. Next, it again picks the link with maximum weight among the remaining links of set $\mathcal{S}$ and eliminates the link causing interference to it. This process is repeated until all links have been considered. The trade-off here is the reduced network capacity. In [13], the authors solve a centralized network throughput maximization problem that uses the backpressure algorithm. The study also implements the solution on hardware to perform evaluation. Even though the network achieves throughput improvement, the network may be prone to last packet problem which is a crucial hindrance for tactical networks. The last packet problem of Q-BP algorithm arises because of the assumption that flows have an infinite amount of data packets being injected into the network. Instead, in practical networks the flows may be finite with some flows terminating and new flows emerging. When a finite flow has the last packet in the queue, it may be stagnant for an extended period of time because of the presence of other queues with larger backlog. This is referred to as the last packet problem. It has been shown that in these cases, queue length based schemes may not be throughput optimal [14]. Accordingly, there has been considerable work on delay-based scheduling [15], [16], [17], [18], [19], [20], [21] to improve the delay performance of the network and eliminate the last packet problem.

In [15], the authors use a shadow queuing architecture so that each node maintains only one queue per neighbor (irrespective of sessions) to reduce the complexity of the queuing structure and improve the delay performance at the cost of throughput. Each node still has to maintain a separate shadow queue (a counter) for every flow going through the node. The backpressure algorithm is executed using the shadow queue counters and these counters are updated according to the optimal number of shadow packets chosen to be transmitted over each link. The key point here is that the number of shadow packets is like a permit to transmit on the given link from the real queue but not associated with the flow of the shadow packet itself. The packet injection rate of the shadow queue is kept slightly higher than the actual packet injection rate. The rates are designed as follows: if the

packet injection rate of the shadow queue is $x_t(t)$, the rate of the real queue is given by $\beta x_t(t)$, where $\beta$ is a positive real number smaller than one. Therefore, if the number of real packets in the queue is less than the number of shadow packets to be transmitted, all the real packets in the corresponding queue are transmitted. The authors show that the real queue length decreases uniformly at every node as the value of $\beta$ decreases, thus leading to lower delays by Little's law. This decrease in delay is accompanied by reduced throughput performance. Maintaining a single queue per neighbor is only beneficial in scenarios where the number of flows through a node is much greater than the number of neighbors. Authors propose a self-regulated MaxWeight scheduling algorithm in [22], where each node estimates the aggregated link rate. They prove that the self-regulated MaxWeight scheduling is throughput-optimal (i.e., stabilize any traffic that can be stabilized by any other algorithm) when the traffic flows are associated with fixed routes and the packet arrivals follow some statistical property. Both [15] and [22] are designed for fixed route scenarios, thus lacking the improvement that could be achieved by opportunistic routing. In [16], the authors propose a delay-limiting algorithm to control the burstiness and delays. They adapt the upper limit for the physical queues to ensure an upper per-hop delay limit at the expense of throughput. To ensure that nodes in the network remain operational, a lower bound has to be set on the upper queue limit. If the traffic reduces to a point such that lower bound comes into play, the delay-limiting approach becomes ineffective. There is also a trade-off between delay and the degree of multipath and opportunism. As the traffic is spread spatially to utilize multiple routes, the lower bound on the queue may again render the delay control ineffective.

A cross-layer design is proposed in [17] using VQ structures to provide finite buffer size or worst-case delay performance. In [18], authors design a delay-aware joint flow control, routing, and scheduling algorithm for multihop network to maximize network utilization. However, due to their ([18], [17]) centralized nature and high complexity they are not well suited for practical distributed implementation [23]. In [19], a throughput optimal scheduling algorithm is proposed using largest weighted delay first algorithm. The idea is to serve the queue $j$ for which $\gamma_j W_j(t) r_j(t)$ is maximal, where $W_j(j)$ is the weighted delay and $r_j(t)$ the achievable capacity for link $j$. Although this algorithm is an easy and distributed way to achieve throughput optimality, this formulation does not take into account the dynamic routing possibilities or queuing dynamics of multihop traffic. Since [19] fails to capture the queuing dynamics of multihop traffic, a new delay metric is defined in [20] to establish a linear relation between queue length and delay. The authors also propose a greedy algorithm that is similar to GSM discussed earlier, but uses delay differential rather than queue length. Simulations show that the average queue length of the network is similar in Q-BP and delay-based backpressure (D-BP) but the tail of the delay distribution is much longer for Q-BP. This implies that some queues are stagnant over extended periods of time in Q-BP whereas D-BP reduces this problem. Unlike the proposed algorithm, D-BP is designed for fixed routes and does not consider dynamic routing. In [21], a delay-driven MaxWeight scheduler is presented that gets around the last packet problem and addresses instability of the queue length based algorithms caused by rate variations. However, it has been shown in [24], [25] that there are other factors that contribute to the inefficiency of the back-pressure algorithm including, inefficient spatial reuse, failure to opportunistically exploit better link rates, underutilized link capacity and inefficient routing because of insufficient path information. Deadline-based routing has been recently studied in [26], [27] and [28]. In [26], an utility-based algorithm is proposed for cyclic mobile social networks under the assumption that nodes follow cyclic mobility, periodically encountering each other with high probability. It is difficult to extend [26] to tactical ad-hoc networks without apriori knowledge of the encounter probability. To increase the packet delivery ratio, [27] adopts an epidemic based routing algorithm and [28] proposes a capacity-constrained routing algorithm that decides which packets have to be replicated. The replication strategies proposed in [27] and [28] to improve packet delivery ratio may adversely affect the achievable throughput.

The major contributions of this work can be outlined as follows,

- We propose a novel deadline-based joint routing and spectrum allocation algorithm for tactical ad-hoc networks to meet the deadline requirements of multiple sessions. To the best of our knowledge, this is the first work that combines the interaction of opportunistic routing, spectrum allocation and deadline constraints to maximize the effective throughput of tactical ad-hoc networks.
- The proposed algorithm is able to adapt to the needs of a dynamic network by managing multiple sessions with variable QoS. This is accomplished by making an optimal choice about the session, route, spectrum and power allocation used to maximize the utilization of available resources.
- A distributed approach is formulated to enable the implementation of the proposed algorithm in a scalable manner.
- Performance of the proposed algorithm is extensively evaluated under various simulated scenarios.
- Another major contribution of this article is the testbed implementation. To prove the practicality of the proposed algorithm, we successfully implement the deadline-based joint routing and spectrum allocation algorithm on a software defined testbed.
- A secondary objective of this paper is to further advance the cross-layer framework and show how novel cross-layer algorithms can be implemented on testbed using the experimental framework.

The rest of the paper is organized as follows. In Section 3, we describe the system model. We discuss the design of deadline-based routing algorithm in Section 4. Next in Section 5, we simulate a 49 node ad-hoc network, to evaluate the performance of the proposed algorithm. The design and configuration of the testbed is described in Section 6. The experimental evaluation of the proposed algorithm on a SDR based testbed is discussed in Section 7. Finally, conclusions are discussed in Section 8. Table 1 summarizes notations used in the paper.

## 3 System Model

Consider a multihop tactical ad-hoc network with $M$ primary users and $N$ secondary users modeled as a directed connectivity graph $\mathcal{G}(\mathcal{U}, \mathcal{E})$, where $\mathcal{U} = \{u_0, u_1, \ldots, u_{N+M}\}$ is a finite set of wireless transceiver (nodes), and $(i, j) \in \mathcal{E}$ represents unidirectional wireless link from node $u_i$ to node $u_j$ (for simplicity, we also refer to them as node $i$ and node $j$).

TABLE 1
Definition of Notations

| Notations | Definition |
|---|---|
| $M$ | Number of primary users |
| $N$ | Number of secondary users |
| $u_i$ | Node $i$ (also refered to as $i$) |
| $\mathcal{PU}$ | Subset of primary users |
| $\mathcal{SU}$ | Subset of secondary users |
| $\mathcal{G}(\mathcal{U},\mathcal{E})$ | Directed connectivity graph, where $\mathcal{U}$ is a finite set of nodes, and $\mathcal{E}$ represents set of unidirectional wireless links |
| $\mathcal{NB}_i$ | Set of neighbors for node $i$ |
| $BW$ | Total available spectrum |
| $[f, f+\Delta B]$ | Set of contiguous frequency bands, where $\Delta B$ is the bandwidth of the cognitive radio |
| $b$ | Bandwidth of each subband |
| $f_{min}$ | Minimum frequency that node can tune to |
| $f_{max}$ | Maximum frequency that node can tune to |
| $s_i \in S_i$ | Set of session in node $i$ |
| $t$ | Time slot |
| $\lambda_i^s(t)$ | Arrival rate of session $s_i$ at $t$ |
| $\Lambda$ | Set of arrival rates |
| $q_i^s(t)$ | A packet in session $s_i$ at $t$ |
| $Q_i^s(t)$ | Virtual queue length of session $s_i$ at $t$ |
| $L(q_i^s)$ | Length of the packet in bits |
| $T_r(q_i^s)$ | Remaining life time of the packet |
| $D(q_i^s)$ | Deadlines of the packet |
| $T_d(q_i^s)$ | Time to the destination of packet as estimated at node $i$. |
| $w_{q_i^s}(L, T_d, T_r)$ | Weight of the packet |
| $R$ | Communication range of the node |
| $T_h$ | Average time spent by packet at each hop |
| $\alpha$ | Delay estimation factor |
| $a(q_i^s, j, t)$ | $a = 1$ represent packet $q_i^s \in Q_i^s(t)$ is transmitted to node $j$ at time slot $t$, and $a = 0$ otherwise. |
| $\mathbf{A}$ | Vector or routing profile |
| $r_{ij}^s(t)$ | Transmission rate for link $(i,j) \in \mathcal{E}$ |
| $\mathbf{R}$ | Vector of transmission rates |
| $\mathbf{P}$ | Selected power levels in each subband |
| $\mathbf{F}$ | Selected frequency subbands |
| $U_{ij}$ | Utility function for link $(i,j) \in \mathcal{E}$ |
| $C_{ij}$ | Achievable channel capacity for link $(i,j) \in \mathcal{E}$ |
| $P_i(f)$ | Transmit power of node $i$ on the frequency $f$ |
| $PL_{ij}(f)$ | Transmission loss due to path loss from $i$ to $j$ |
| $G$ | Processing gain |
| $N_j(f)$ | Receiver noise on frequency $f$ |
| $I_j(f)$ | Interference experienced by the receiving node $j$ |
| $BER_{\mathcal{PU}}$ | BER guarantees required for primary user |
| $BER_{\mathcal{SU}}$ | BER guarantees required for secondary user |
| $SINR_{\mathcal{PU}}^{th}$ | SINR thresholds required to achieve the target $BER$ for the primary user |
| $SINR_{\mathcal{SU}}^{th}$ | SINR thresholds required to achieve the target $BER$ for the secondary user |
| $O_{ij}(f)$ | Spectrum Opportunity between $i$ and $j$ |
| $P_i^{max}(f)$ | Maximum power that can be used by the secondary node $i$ on the frequency $f$ |
| $P_i^{min}(f)$ | Minimum power required to reach the required $SINR_{\mathcal{SU}}^{th}$ at the intended secondary receiver |
| $CW$ | Contention window |
| $\eta$ | Effective throughput |
| $\rho$ | Reliability |

We assume $\mathcal{G}$ is link symmetric, i.e., if $(i,j) \in \mathcal{E}$, then $(j,i) \in \mathcal{E}$. The nodes from the subset $\mathcal{PU} = \{u_1, \ldots, u_M\}$ are designated as primary users, and nodes from the subset $\mathcal{SU} = \{u_{M+1}, \ldots, u_{M+N}\}$ are designated as secondary users. The secondary network is composed of cognitive nodes capable of adapting to the current spectrum usage. The primary user holds the license for the specific spectrum bands and have full access to the spectrum without interference from any other users. In relevant scenarios, the primary user can also be a non-cooperative node (the adversary). Since the entire spectrum is not always used by primary users, the aim of the secondary user in a cooperative scenario is to maximize spectrum utility while ensuring no interference to primary users. Thus, a secondary user has to use the spectrum holes [2] to maximize the spectrum usage. The secondary network will also allocate resources such that it maximizes the number of packets delivered at the destination within their respective deadline. Only packets that reach the destination within the specified deadline contribute towards the effective throughput computation. The set of neighbors for node $i$ is given by $\mathcal{NB}_i \triangleq \{j : (i,j) \in \mathcal{E}\}$.

The secondary users are equipped with cognitive radios capable of scanning the available spectrum to reconfigure their transceivers on-the-fly. The entire available spectrum is given by $BW$. The cognitive transceiver is capable of tuning to a set of contiguous frequency bands $[f, f+\Delta B]$, where $\Delta B$ is the bandwidth of the cognitive radio and $\Delta B < BW$. We also assume that the transmit power can be varied to exploit any available spectrum opportunity. We define spectrum opportunity as the limited availability of spectrum that might currently be used by nodes (primary or secondary users) but can be further exploited by adjusting the transmit power such that it does not violate the bit error rate (BER) constraint of the existing transmission. This work is intended for any general physical layer but we assume that multiple transmissions can occur concurrently on the same frequency band, e.g., with different spreading codes.

The total spectrum, $BW$ is divided into separate channels, a common control channel (CCC) and a data channel. All secondary nodes use CCC to share local information for spectrum negotiation and data channel is used exclusively for data communication. The data channel is divided into discrete set of carriers $\{f_{min}, f_{min+1}, \ldots, f_{min-1}, f_{max}\}$, each of bandwidth $b$ and identified by a unique discrete index. The cognitive radio of the secondary user can tune into a consecutive set of carriers from $[f_{min}, f_{max}]$. Let the traffic in the network consist of multiple sessions characterized by the source-destination pair and the application generating the session. The arrival rates of each session $s_i \in S_i$ at node $i$ is given by $\lambda_i^s(t)$, and characterized by vector of arrival rates $\Lambda$.

## 4 Deadline-Based Routing and Spectrum Allocation

In this section, we discuss the deadline-based distributed routing and spectrum allocation algorithm in detail. Here, we will define the utility function, that has to be maximized to achieve the goal of the proposed solution.

### 4.1 Network Utility Function

Consider that the tactical ad-hoc network is assumed to operate over a time slotted channel. The spectrum utility function is calculated by node $i$ for every time slot $t$ when node $i$ is backlogged and not already transmitting or

receiving packets. Each node $i$ maintains a separate VQ for each session. We define $Q_i^s(t)$ as the VQL formed by packets of session $s$ in node $i$ at time slot $t$. Unlike traditional queue length, the VQL gets inflated as time passes to penalize the node for holding packets whose deadline is approaching. More details about the design of VQL is discussed below. For each packet $q_i^s \in Q_i^s(t)$, that belongs to session $s$ and stored at node $i$, a set of fields are defined, including,

- $L(q_i^s)$ is the length of the packet in bits,
- $T_r(q_i^s)$ is the remaining life time of the packet, which is based on the deadline $D(q_i^s)$ assigned to the packet at the source node,
- $T_d(q_i^s)$ is the time to the destination as estimated at node $i$.

Based on these parameters, a weight $w_{q_i^s}[L(q_i^s), T_d(q_i^s), T_r(q_i^s)]$ can be defined for each packet $q_i^s \in Q_i^s(t)$ as follows,

$$w_{q_i^s}(L(q_i^s), T_d(q_i^s), T_r(q_i^s)) = \frac{L(q_i^s)}{max(T_r(q_i^s), \tau) max(T_r(q_i^s) - T_d(q_i^s), \tau)}. \tag{1}$$

As we can see in (1) the weight $w_{q_i^s}$ assigned to each packet is directly proportional to $L$ (for simplicity, we removed $q_i^s$ from these notations) and inversely proportional to $T_r$ and $T_d$. The $\tau$ in (1) is a very small value used to avoid negative and infinite weights. The parameter $T_r$ helps to get rid of the well-known last packet problem, since $T_r$ will increase the VQL as time elapses. This can be interpreted as the holding penalty imposed for packets being stagnant in the queue for extended period of time. Since $T_r$ is dependent on the assigned deadline, it helps the nodes to manage different sessions by pushing critical packets faster even if the actual queue length is comparatively smaller. Considering just the deadlines alone will not help in cases where there are two sessions with the same deadline but one is farther away from the destination than the other. In such cases, $T_d$ will ensure that the session farther away from the destination moves through the network at a faster rate compared to similar sessions closer to the destination. Therefore, $T_d$ can be considered as a variable that either amplifies or diminishes the effect of $T_r$ depending on the time required to reach the destination. $T_d$ also encourages packets to take shorter routes if all other factors like queue length and spectrum are the same for two different routes. The rationale will become more evident when we discuss the network utility function used for the proposed algorithm.

Among these three parameters, the exact value of $T_d$ is not available at each node and has to be estimated at each hop. For a centralized network, assuming global knowledge of the network, $T_d$ can be estimated using average queuing delays, transmission rate, propagation delays and using the knowledge of average delays experienced previously by packets with the same destination. Estimating $T_d$ becomes further challenging in a distributed network where each node is required to make decisions without global knowledge of the network. One solution is to estimate $T_d$ by using queuing delay experienced by the session in the node itself. We use this information and slightly over estimate the delay by assuming that the packet has to route through more than one node within its transmission range itself. Underestimating $T_d$ would increase the risk of packets not reaching the destination within the specified deadline. Therefore, in our design $T_d$ is slightly over estimated according to the characteristics of the network. Since $T_d$ is updated at every hop, the estimation error/margin decreases as the packet moves closer to the destination. This method does not lead to any error propagation since the value is updated at each hop. A simple way to estimate $T_d$ is based on distance to destination $(d)$, communication range (estimated based of maximum transmit power) of the nodes deployed $(R)$ and average time spent by the packet during each hop $(T_h)$ (estimated based processing delay, queuing delay, transmission delay and propagation delay). The idea is to assume that a hop is required every half range of a node and is given by $\alpha = R/2$. Accordingly, we get an estimate of how much time is required to reach the destination as,

$$T_d = \frac{d\, T_h}{\alpha} = \frac{2d\, T_h}{R}. \tag{2}$$

The value of $\alpha$ can be varied according to the density of the network. Now from the definition of weights, it can be seen that higher value is assigned to packets with more bits to transmit, lower $T_r$ and which are farther away from the destination. Accordingly, we define a VQL of a session $s$ in node $i$ as follows,

$$Q_i^s(t) = \sum_{q_i^s \in Q_i^s(t)} w_{q_i^s}(L, T_d, T_r). \tag{3}$$

Now, let $a(q_i^s, j, t) = 1$ represent a packet $q_i^s \in Q_i^s(t)$ is transmitted to node $j$ at time slot $t$, and $a(q_i^s, j, t) = 0$ otherwise. The routing profile of node $i$ is defined as $a_i^s(t) = [a(q_i^s, j, t)]_{q_i^s \in Q_i^s(t)}^{j \in \in \mathcal{SU}/i}$, and $\mathbf{A}$ represents the vector of routing profile $a_i^s(t)$ of all nodes in the network at instant $t$. We also define the transmission rate on link $(i, j)$ during time slot $t$ as $r_{ij}^s(t)$, and $\mathbf{R}$ as the vector of rates. Then, the VQL of node $i$ can be updated as,

$$Q_i^s(t+1) = \Bigg[ Q_i^s(t) + \sum_{j \in \mathcal{N}/i} \sum_{q_j^s \in Q_j^s(t)} w_{q_j^s}(L, T_d, T_r) a(q_j^s, i, t) - \sum_{j \in \mathcal{N}/i} \sum_{q_i^s \in Q_i^s(t)} w_{q_i^s}(L, T_d, T_r) a(q_i^s, j, t) \Bigg]^+. \tag{4}$$

Accordingly, the network link utility function $U_{ij}$ for link $(i, j) \in \mathcal{E}$ for session $s$ can be defined as,

$$U_{ij}(a_i^s(t)) = C_{ij}[Q_i^s(t) - Q_j^s(t)]^+, \tag{5}$$

where $[Q_i^s(t) - Q_j^s(t)]^+$ represents the differential VQL and $C_{ij}$ is the achievable channel capacity of the link $(i, j) \in \mathcal{E}$ at time slot $t$ for a selected frequency $(f)$ and the transmission strategy can be given by,

$$C_{ij}(f, P_i(f)) \triangleq \sum_{f \in [f_i, f_{i+\Delta f_i}]} b.\log_2 \left[ 1 + \frac{P_i(f) PL_{ij}(f) G}{N_j(f) + I_j(f)} \right]. \tag{6}$$

In the above equation, $P_i(f)$ represents the transmit power of node $i$ on the frequency $f$, $PL_{ij}(f)$ is defined as the transmission loss due to path loss (can be computed based on the chosen path loss model) from $i$ to $j$, $G$ represents the processing gain, which would be the length of the spreading code when applicable, $N_j(f)$ is the receiver noise on frequency $f$ and $I_j(f)$ is the interference experienced by the receiving node $j$. We assume a quasi-static channel, i.e., channel conditions remain constant for the duration in between sensing and transmission of a packet. This can be

achieved with an efficient sensing mechanism and having dedicated receiver that perform sensing in parallel to the regular transceiver. As we can see in (6), the achievable capacity primarily depends on selected frequency $\mathbf{F} = [f_i, f_{i+\Delta f_i}]$, power allocation $\mathbf{P} = [P_i(f)]$, $\forall i \in \mathcal{SU}, \forall f$ and the scheduling policy. Therefore, the overall notion of this network utility function is to couple the constraints of packet deadline to the traditional queue length used in the differential backlog algorithm. This is then weighted by the dynamic spectrum availability information to provide a joint routing and spectrum allocation decision. Moreover, algorithms like ROSA [29], [30] does not handle QoS requirements of different traffic classes. Since this is essential for improving the reliability of tactical ad-hoc networks, the redefining of the the queue length to form the new VQL is where the proposed algorithm extends [2]. We will discuss and evaluate the benefits of this in detail in Section 5.

### 4.2 Distributed Deadline-Based Routing and Spectrum Allocation Algorithm

The overall optimization problem is to maximize the utility function discussed in (5). Let us denote the BER guarantees required for primary and secondary users as $BER_{\mathcal{PU}}$ and $BER_{\mathcal{SU}}$ respectively. Accordingly, we can represent the required signal-to-interference-plus-noise power ratio (SINR) thresholds required to achieve the target $BER$ for the secondary and primary user as $SINR^{th}_{\mathcal{PU}}$ and $SINR^{th}_{\mathcal{PU}}$ respectively. Thus, the global objective of the optimization problem is to find the optimal global vectors **R**, **F** and **P** that will maximize the sum of the network utilities, under the power and $BER$ constraints. The formulation of the optimization problem is provided in Appendix. Since solving the overall optimization problem needs global knowledge of feasible rates and the worst-case complexity of this centralized problem is exponential, it necessitates the need to design a distributed algorithm that is scalable for practical implementation.

The resource allocation of the proposed algorithm consists of spectrum and power allocation. A spectrum opportunity for link $(i, j)$ is a set of contiguous subbands where $O_{ij}(f) \geq 0$, when $O_{ij}(f)$ is given by,

$$O_{ij}(f) = P_i^{max}(f) - P_i^{min}(f), \tag{7}$$

where $P_i^{max}(f)$ is defined as the maximum power that can be used by the secondary node $i$ on the frequency $f$ such that it satisfies the BER constraints of primary and secondary users. It is important to note that $P_i^{max}(f)$ will be constrained by the maximum transmit power of the wireless radio used in the network. On the other hand, $P_i^{min}(f)$ denotes the minimum power required to reach the required $SINR^{th}_{\mathcal{SU}}$ at the intended secondary receiver. In other words, $P_i^{min}(f)$ and $P_i^{max}(f)$ provide the lower bound and upper bound of transmit power respectively for node $i$ on frequency $f$. The $P_i^{min}(f)$ and $P_i^{max}(f)$ values are determined by a node $i$ by gathering spectrum and resource allocation information from its neighbors. This information is gathered using collaborative virtual sensing (CVS) using the control packets in the network. We do not include details about resource allocation, CVS and the medium access control (MAC) protocol employed as it is similar to that in ROSA. We urge readers to refer to [2] for further details.

Accordingly, we propose the distributed Deadline-based cross-layer Routing and Spectrum allocation algorithm (DRS) to maximize the throughput of a tactical ad-hoc network. In the distributed network, each node makes an adaptive decision to choose optimal session, next hop, power allocation and spectrum to use during the next time slot based on the information gathered from the neighbors using CVS. This decision will be different from traditional ROSA [2] because the network utility defined here is a function of VQL and not the actual queue lengths. Once a backlogged node senses an idle CCC, it performs the Algorithm 1 to obtain the optimal resource allocation decision:

**Algorithm 1.** Deadline-based Resource Allocation

$t = 1, \Delta = \infty, C_{ij} = 0, U^*_{ij} = 0$
**for** $s_i \in S_i$ **do** ▷ All Active sessions
  **for** $j \in u_1, u_2, ...u_k$ **do** ▷ Next feasible hops
    **for** $f_i \in [f_{min}, \ldots, f_{max-\Delta f_i}]$ **do**
      Calculate $P_i^t(f)$ similar to [2]
      Calculate $C_{temp}$ as in (6)
      **if** $C_{temp} > C_{ij}$ **then**
        $C_{ij} = C_{temp}$
        $[f^*_{i,j}, \mathbf{P^*_{i,j}}] = [f_i, \mathbf{P^t_i}]$
      **end if**
    **end for**
    $U^s_{ij} = C_{ij} * [Q^{s_i}_i - Q^{s_i}_j]$
    **if** $U^s_{ij} > U^*_{ij}$ **then**
      $U^*_{ij} = U^s_{ij}$
      $[f_i^{opt}, \mathbf{P_i^{opt}}, s_i^{opt}, j^{opt}] = [f^*_{i,j}, \mathbf{P^*_{i,j}}, s_i, j]$
    **end if**
  **end for**
**end for**
Return $[f_i^{opt}, \mathbf{P_i^{opt}}, s_i^{opt}, j^{opt}]$

(1) The proposed algorithm assumes that location of the intended destination node is known to the source node. This information is carried by the packet through the intermediate nodes. Each node selects a feasible set of next hops for each backlogged session $j \in (u^s_1, u^s_2, \ldots, u^s_k)$, which are neighbors with positive advance towards the intended destination.

(2) The maximum capacity for each node is calculated by considering all possible spectrum opportunities. The maximum capacity of each feasible neighbor is used along with the corresponding differential VQL to determine the network utility $U^s_{ij}$. The optimal decision is taken such that,

$$(s^{opt}, j^{opt}) = \arg\max(U^s_{ij}). \tag{8}$$

As seen earlier, the network utility function comprises of differential VQL and achievable capacity. The differential VQL is a function of deadline and estimation of $T_d$. Thus, the sessions that have smaller deadlines or are further away from the intended destination will be scheduled more often if the available spectrum for all sessions are comparable. The adaptive routing will also provide most traffic to VQs that are lightly backlogged.

(3) The optimal frequency and power allocation $(f_i^{opt}, \mathbf{P_i^{opt}})$ correspond to the values that provide maximum Shannon capacity $C_{ij}$ over the wireless link $(i, j^{opt})$, where $j^{opt}$ is the best next hop.

TABLE 2
Parameters of Scenario 1

| Parameter | Experiment 1 | Experiment 2 |
|---|---|---|
| Source Rate | 2 Mbits/s | 2 Mbits/s |
| Session duration | 5 s | 5 s |
| Session start | randomly from $t = [0, 5]$ s | randomly from $t = [0, 5]$ s |
| No. of sessions | 2, 4, 6, 8, 10, 12 14, 16, 18, 20, 22 | 2, 4, 6, 8, 10, 12 14, 16, 18, 20, 22 |
| Deadline of each session | 2 s | Odd session 1.5 s Even session 10 s |

In this work, we use a contention based medium access control protocol in the control channel before transmitting the packet on the selected data channel. In the contention based MAC protocol, the probability of accessing the medium is calculated based on the $U_{ij}^{*}$. Nodes generate a backoff counter from the range $[0, 2^{CW-1}]$, where $CW$ is the contention window. The $CW$ is a decreasing function of $U_{ij}^{*}$. This will ensure that heavily backlogged VQs with more spectrum resources will have a higher probability of transmission.

The computational complexity of the DRS algorithm at a node $i$ is directly proportional to the number of neighbors, number of channels and number of active sessions. Therefore, for a constant number of channels and sessions in a network the computational complexity for node $i$ is given as $\mathcal{O}(|\mathcal{NB}_i|)$.

## 5 Performance Evaluation through Simulation

In this section, we compare the performance of DRS with ROSA in a multihop ad-hoc network. To evaluate DRS, we use a object-oriented packet-level discrete-event simulator similar to [2], which implements the features described in the earlier sections of this paper. The metric used for this evaluation is effective throughput ($\eta$) and reliability ($\rho$) of the network. Effective throughput was defined based on the number of packets received within the deadline. The reliability is defined as the ratio of packets received at the destination within the specified deadline with respect to the number of packets generated at the source node. The evaluation is conducted on a grid topology in a 6000 m x 6000 m area. The sessions are initiated between disjoint random source-destination pairs and the packet size of the packets are set at 2500 bytes and number of packets transmitted per session is set to 500. The total available spectrum ($BW$) is set to be 54 MHz-72 MHz The bandwidth usable by cognitive radios are restricted to be 2, 4 and 6 MHz. The bandwidth of the common control channel is set as 2 MHz. Each result was obtained by averaging the values obtained from 50 random seeds unless specified differently. Next, we describe different scenarios under which the proposed algorithm is compared to ROSA. In all figures except Fig. 6, the blue lines represent performance of DRS and red lines denote the performance of ROSA.

### 5.1 Scenario 1: Network Performance As the Number of Session Increases (All Sessions Started at Random Time)

In scenario 1, we evaluate the network performance as the number of active sessions in the network increase. The parameters used during the two experiments for scenario 1 are listed in Table 2. The only difference between the two experiments are the deadlines assigned to different sessions. In experiment 1, all the sessions have a deadline of 2 s, which represents a highly constrained network. Instead, in experiment 2 the odd numbered sessions have a deadline of 1.5 s and even numbered sessions have a deadline of 10 s. Experiment 2 can be considered as a scenario where one session carries periodic weather monitoring data through the network. These sessions are delay tolerant to an extent, hence have a longer deadline. The second type of data can have extremely small deadline, consisting of delay-intolerant data like threat detection, incoming missile alert or real-time video streaming. The proposed algorithm should be able to adapt to the varying requirements of different sessions and maximize the effective throughput of the network. The session are set to start randomly any time between start of the simulation ($t = 0$ s) and session duration ($t = 5$ s). This ensures that all sessions are active at some point during the simulation but the number of active sessions will vary throughout the simulation. The parameters of both the experiments (1 and 2) are listed in Table 2. Examining Figs. 2 and 3 show that DRS performs much better than ROSA in terms of reliability and effective throughput in experiment 3 and 4. In these scenarios, traditional backpressure based algorithm may suffer from the last packet problem. Since DRS is formulated based on VQL which takes into account the deadlines of each packet in the queue, the penalty for holding packets in the queue grows as time elapses eliminating the last packet problem.

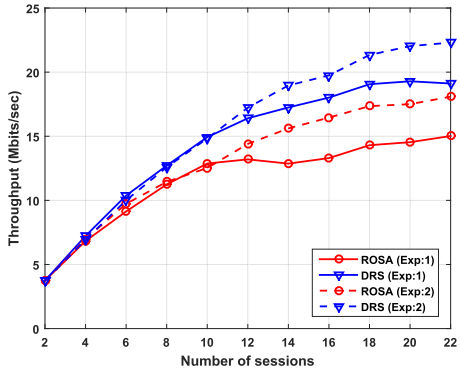


Fig. 2. Scenario 1: $\eta$ versus No. of sessions.

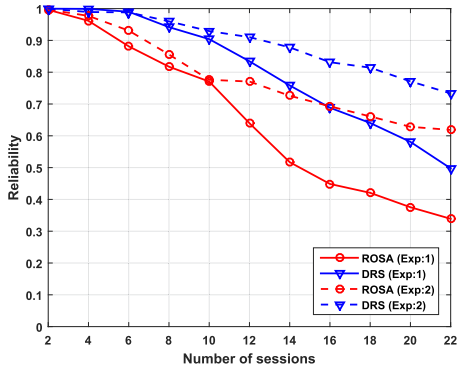


Fig. 3. Scenario 1: $\rho$ versus No. of sessions.

### 5.2 Scenario 2: Network Performance As the Data Rate of the Sessions Increase

In these set of experiments, we evaluate the network performance in a scenario where number of sessions are kept

TABLE 3
Parameters of Scenario 2

| Parameter | Exp: 3 | Exp: 4 | Exp: 5 |
|---|---|---|---|
| Source Rate | 1 to 10 Mbits/s | 1 to 10 Mbits/s | 1 to 10 Mbits/s |
| Session duration | 5 s | 5 s | 5 s |
| No. of sessions | 5 | 5 | 5 |
| Deadline of each session | 2 s | Odd session 1.5 s<br>Even session 10 s | 2 s |

TABLE 4
Parameters of Scenario 3

| Parameter | Exp: 6 | Exp: 7 | Exp: 8 |
|---|---|---|---|
| Source Rate | 2 Mbits/s | 2 Mbits/s | 2 Mbits/s |
| Session duration | 5 s | 5 s | 5 s |
| No. of sessions | 5 | 8 | 2, 4, 6, 8, 10, 12, 14, 16 |
| Deadlines | 2 s | 2 s | 50 s |
| No. of seeds | 50 | 50 | 20 |

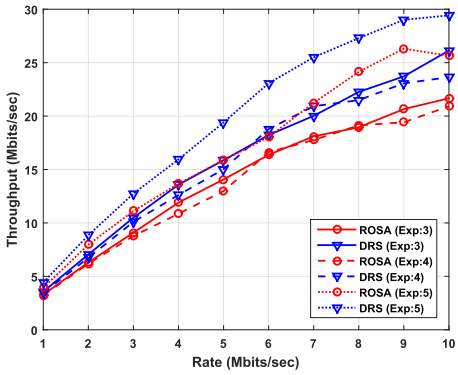


Fig. 4. Scenario 2: $\eta$ versus No. of sessions.

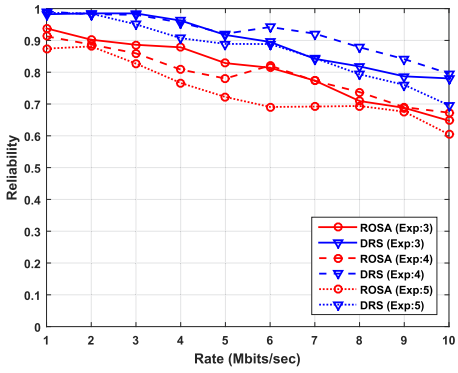


Fig. 5. Scenario 2: $\rho$ versus No. of sessions.

constant and the data rate injected at the source node increases from 1 Mbits/s to 10 Mbits/s. We conduct three experiments in this scenario. Experiments 3 and 4 are similar to the experiments of previous scenario 1, varying only in deadline as shown in Table 3. Experiment 7 evaluates the performance of DRS when the packet size is larger (25000 bits). As we can see in Figs. 4 and 5, all three experiments show that DRS outperforms ROSA. Though the effective throughput of both ROSA and DRS increased with higher packet size (dotted lines), the reliability of ROSA decreased more compared to the decrease in DRS. Hence, the difference in performance between DRS and ROSA increased when larger packets (fewer number of packets per second) were used in the network.

### 5.3 Scenario 3: Examining the Effect of Different Components of DRS

Here, we try to evaluate the effect of different components used during the formulation of DRS. In experiment 6, we evaluate how $T_r$ and $T_r - T_d$ affect the proposed algorithm individually. Accordingly, we run the simulation with the

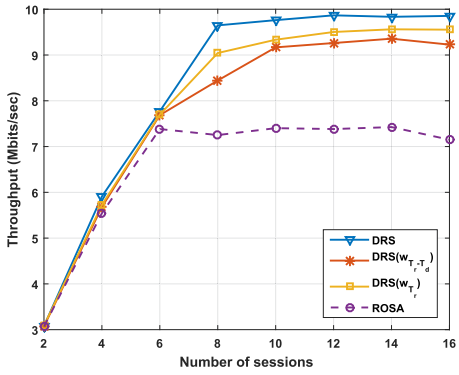


Fig. 6. Scenario 3: $\eta$ versus No. of sessions.

parameter shown under experiment 6 in Table 4 using two different weight definitions as shown below,

$$w_{T_r} = \frac{L}{max(T_r, \tau)} \tag{9}$$

$$w_{T_r - T_d} = \frac{L}{max(T_r - T_d, \tau)} \tag{10}$$

Fig. 6 shows that both the DRS using weights seen in (9) and (10) perform considerably better than ROSA but does not maximize the effective throughput like original DRS. This is because original DRS that uses the weight shown in (1) derives the benefits of both weights ((9) and (10)). Hence, this shows why it is advantageous to have both $T_r$ and $T_r - T_d$ in the denominator of the weight used to calculate VQL. Further, it is interesting to note that in cases where it is difficult to estimate $T_d$, one can still achieve moderately good performance by using weight shown in (9). Next, we evaluate the effect of the parameter $\tau$ on the effective throughput of the network. Equation (1) uses a very small value $\tau$ to ensure correctness of weight, such that instances with infinite value do not occur.

Fig. 7 depicts the effective throughput of the network as $\tau$ changes while keeping the number of sessions and source data rates constant. The other parameters of experiment 7 are depicted in Table 4. The result shows that the effective throughput of the network using DRS is consistently high for values of $\tau$ over a range between $10^{-8}$ to 0.99. Any value greater than $10^{-1}$ takes away the effect of deadlines and has a degrading effect on effective throughput. As the value of $\tau$ moves closer to 1, the VQL becomes more and more equivalent to traditional queue length. On the other hand, choosing $\tau$ to be smaller than $10^{-8}$ also affects the algorithm adversely since it bloats the VQL to an extent where the capacity component of the network utility function becomes insignificant. This lower bound would depend on the characteristics of the network, specifically, the achievable

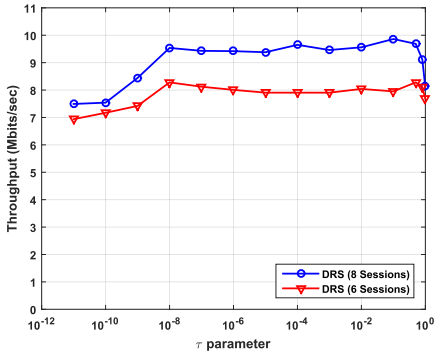


Fig. 7. Scenario 3: $\eta$ versus Parameter $\tau$.

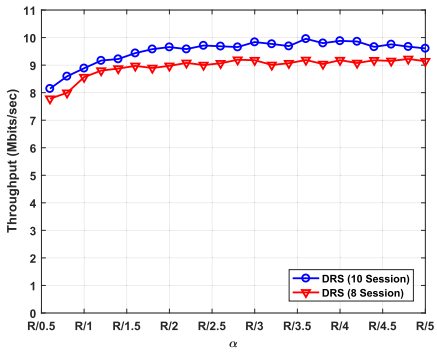


Fig. 8. Scenario 3: $\eta$ versus Parameter $\alpha$.

capacity determined by the bandwidth of the transceiver. Fig. 7 also shows that the range of values for $\tau$ outside which the effective throughput of the network starts declining is same for both cases (6 and 8 sessions). This shows that the acceptable value for $\tau$ does not change according to the number of active sessions in the network. Since we have shown that DRS performs consistently well over a large range of values of $\tau$, one can choose any value within the acceptable range depending on the network setup.

Next, we use the same parameters as in experiment 7 to analyze how errors in estimation of $T_d$ affect the network's effective throughput. In this case, we set the number of sessions to eight and ten. We vary $\alpha$ from $R/0.5$ to $R/5$ as shown in Fig. 8. When $\alpha = R/0.5$, $T_d$ is underestimated and when $\alpha = R/5$, $T_d$ is overestimated. As expected, if $T_d$ is underestimated, fewer packets are delivered to the destination within the deadline thereby decreasing the effective throughput. Meanwhile, overestimation does not impact the throughput negatively because $T_d$ is calculated at each hop and hence error/margin decreases as the packet moves closer to the destination.

Finally, in experiment 8, we evaluate the performance of DRS in a network having sessions with very long deadline ($50$ s). This experiment examines how the network behaves in scenarios where the deadlines of the sessions are long enough such that packets lost due to expiration of deadline are negligible. This experiment evaluates whether there is any loss in throughput while using DRS as compared to ROSA in network that are delay tolerant (have extremely long deadlines). As we can see from Fig. 9, the throughput of both algorithms are equal as the number of sessions in the network increase. This shows that there is no disadvantage in using DRS over ROSA even in scenarios where deadlines are insignificant.

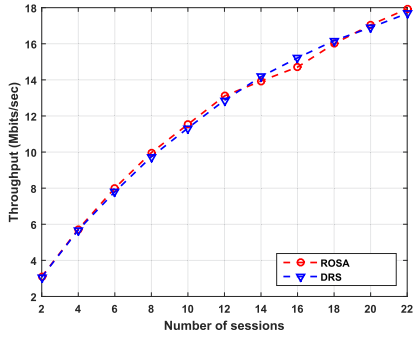


Fig. 9. Scenario 3: $\eta$ versus No. of sessions (Deadline = $50$ s).

## 6 Testbed Implementation

### 6.1 Challenges

In the past decade, cross-layer protocols have been extensively studied and various solutions have been proposed [31], [32], [33], [34]. Even with these advances in literature, most of the solutions are evaluated only using simulation tools like MATLAB, ns-3, OPNET among others. The goal of cross-layer optimization techniques are to utilize the information between different layers and enable their interaction to jointly optimize objectives including throughput, reliability, delay among others. This invokes the need for an architecture that enables these interactions and promotes design and development of cross-layer optimization algorithms. Lack of such platforms has lead to the growing gap between number of solutions proposed in literature versus the number of solutions that are implemented and tested using actual hardware. There are only limited efforts that extend the implementation of the optimization algorithms to actual hardware and evaluate the performance on a cross-layer testbed [13], [35], [36], [37], [38], [39]. The major challenge in achieving this implementation is the lack of a flexible architecture that facilitates implementation of the cross-layer optimization algorithm on multi-node networks. This deficiency is being recognized by the community and some solutions are being proposed to achieve the required flexibility.

GNU radio is an open-source signal processing software that provides great flexibility specifically at the physical layer of SDRs. GNU radio comprises of various signal processing and digital communication blocks and is an excellent tool to control SDR. However, the majority of the contribution is limited only to the Physical layer. There have also been efforts to relocate some of the processing functions to a Field-programmable gate array (FPGA) [40] to improve the delay performance. This makes it difficult to integrate new algorithms for testing and evaluation purpose. Some other work [41], [42] aims to provide reconfigurable MAC protocols by decomposing the overall design into core fundamental blocks. In [41], the implementation of these fundamental blocks are split between PC and FPGA depending on the time critical nature of the blocks. In [42], the authors implement an abstract execution machine on a resource-constrained commodity WLAN (wireless local area network) card. Recently, software defined network (SDN) using an Open-Flow [43] based approach has been proposed for evaluating routing protocols. The overall concept of Open-Flow is to keep the data path on the Open-Flow switch itself while moving the high-level routing decision to a separate controller (server). The switch performs the packet forwarding based on the flow table defined by the

controller and use Open-Flow protocol to communicate with each other. The majority of the work on OpenFlow has been concentrated at the network layer of the protocol stack.

Even with these advancements, a major challenge to transitioning algorithms and protocols to commercial hardware is the lack of a software defined testbed with a flexible architecture that enables easy implementation of cross-layer technologies. These testbeds are essential to corroborate the results obtained in simulations and evaluate how to refine these algorithms to ensure a successful transition to relevant hardware. Some of the requirements of such a testbed include, a flexible cross-layer based protocol stack [38], modularity to integrate new algorithms with ease, a framework to accommodate both centralized and distributed solutions, real-time network performance monitoring tools, and having the ability to run unsupervised scripted experiments over extended periods of time. Having such a testbed expedites the design and development process of next-generation wireless communication technologies destined for a commercial SDR system. The CrOss-layer Based testbed with Analysis Tool (COmBAT) first introduced in [39] was developed to serve as a software defined testbed to enable the implementation of cross-layer optimization algorithms. In COmBAT, a Adaptive cross-layer (AXL) communication framework facilitates easy integration of new protocols and algorithms. The design of AXL is discussed in detail in the next section.

### 6.2 Adaptive Cross-Layer (AXL)

The overall AXL framework depicted in Fig. 11 consists mainly of the application layer, session manager, decision plane, control plane, register plane and the physical layer. Each node in the network uses the AXL framework in place of a traditional protocol stack and are therefore referred to as AXL nodes throughout this paper. In implementation, the AXL framework consists of Python multiprocessing processes which are initialized at node start up using an AXL daemon. The daemon imports the main modules and properties that are used in the different layers/planes of the framework as required. The properties include predefined values for the network such as data timeout duration used by MAC protocol, node IP and MAC addresses and payload sizes. However, most of the properties are dynamic in nature and they can be reconfigured on-the-fly based on network optimization strategies or user input. The processes that are started by the daemon run continuously until shutdown. These processes include the register plane, session manager, control plane and the physical layer. The decision plane is not a process but a collection of functions that can be called by the framework when needed. Each plane/layer can share information with each other by a combination of three methods; direct function calls, shared memory (register plane) or by overhearing global events (global with respect to the framework, not the entire network) that can be triggered by any process in the framework. These functionalities allow for a flexible cross-layer communication between all network protocols.

*Application (APP) Layer.* The current AXL software package provides data generation APPs that a user may choose from in order to evaluate the performance of the network. The APPs can operate in packet streaming mode or packet-by-packet mode. For streaming mode, the source data is repeated until a user specified amount of data has been generated. The streaming mode is generally used in experiments requiring a constant bit rate (CBR) source for a fixed duration of time. The APPs connect to the AXL daemon via a TCP/IP socket. For each APP that connects, a unique connection object is created that manages data transfer between the APP and the AXL framework. Each packet contains the user generated and QoS parameters. This is where deadline of the packet can be defined. The packet is parsed and then sent to the session manager for the next processing stage.

*Session Manager.* In the AXL framework, the session manager provides the capability of simultaneous multi-session management. When a packet arrives at the session manager, the session manager creates a session object based on the packet parameters, which include process ID that is created by the OS, source and destination IP, data type, any QoS parameters (such as deadline), and the packet number generated at packet creation. Packets that correspond to existing session objects are appended to their appropriate session queue. Packets receive their network headers based on the parameters in the session object mentioned earlier. In implementation, the session manager is designed as a multiprocessing first-in, first-out (FIFO) with a user specified update period. The update period dictates the timeout for updating packet queues. During each update period, the session manager stores the current queue length of each session in the register plane and triggers an event flag which indicates that the transmitter has backlogged session ready for routing decisions.

*Decision Plane.* As the name suggests, this is the component where all the logical decision making and algorithm execution takes place. These algorithms pertain to routing algorithms, spectrum allocation, automatic modulation classification and other resource allocation decisions. The complexity of the algorithms can vary from threshold decision to iterative algorithms like the Expectation and Maximization (EM) algorithms or solvers for convex optimization problems. Decision algorithms can (i) modify a parameter in a protocol, (ii) trigger switching among different modes within a protocol, (iii) enable switching among different protocols altogether [38].

In regards to this paper, there are currently two routing algorithms (ROSA and DRS), stored as software modules, available for use (discussed in detail in Section 7). This is where all the calculation for routing algorithm takes place. Each routing module can be passed as an instance for the decision plane to use during runtime. The chosen and active routing module is requested by the session manager to execute the algorithm when a session is backlogged. The results of the executed algorithm is stored in the register plane for other layers/planes to access. Conversely, to execute the algorithm, the decision plane obtains the information from the register plane. After executing the algorithm within the routing module, the decision plane triggers an event flag that prompts the control plane to schedule a transmission. It should be noted that the interactions between the decision plane and the other layers/planes in AXL take place via the register plane, through global events or through direct function calls.

There are some DRS specific requirements that had to be included in the experimental framework to ensure successful implementation. Each packet is required to carry time of generation in its header. This enables nodes to calculate VQL at each hop. This feature had to be included in the experimental framework to enable the operation of such deadline-based cross-layer algorithm. In this work, we included the ability to track the time instant of generation and arrival of packets at each node including the destination. In traditional network, the backlog does not change with time unless a packet is received or transmitted by the node. In case of DRS, the

VQL changes continuously with time. This was a crucial component necessary to implement DRS-like protocols that aim to eliminate the last packet problem. Therefore, in this current version of the framework, the VQL is constantly updated at the decision plane of each node. All of these inclusions have bolstered the experimental environment/framework to handle algorithms that are required to handle time-based queue lengths.

*Control Plane.* The control plane houses the control logic used to access the wireless medium. The control plane contains the finite state machine (FSM) used to implement different MAC protocols. The chosen MAC protocol defines the exact set of states, events, conditions and actions required to operate FSM. The control plane can be initialized to use multiple different MAC protocols depending on the situational awareness gathered from other layers/planes of the stack as shown in [38]. Each MAC protocol should have its FSM implemented in the control plane as a separate FSM initialization function. Future developers can take advantage of the baseline FSM model that is already defined in the control plane by modifying its states and actions as needed by the protocol. An example of a state transition diagram for a carrier sensing multiple access with collision avoidance (CSMA/CA) based MAC protocol [2] is given in Fig. 12. We would like to point out that CSMA/CA based MAC protocol operates only on the CCC and does not restrict concurrent feasible (in terms of BER constraints) transmission from occurring on the data channel. The state transition diagram describes the interaction between all possible states, events and actions for the receive and transmit paths. As shown in Fig. 12, when an event $Data\_available$ is set, $Send\_RTS$ (request-to-send) action is taken as the FSM goes from an $IDLE$ state to $WAIT\ CTS$ (clear-to-send) state. The next event that the FSM is looking for is either $CTS\_received$ or $CTS\_timeout$ and the FSM transitions depending on which event was observed. If $CTS$ was received and $CTS\_received$ is set, $Send\_DTS$ (Data Transmission reServation) action is taken as the FSM goes from $WAIT\ CTS$ state to $SEND\ DATA$ state. The rest of the state transition diagram can be interpreted in a similar manner.

The FSM is generally in an IDLE state until the corresponding global AXL events are flagged to invoke a state transitioning process. These global AXL events are used in cross-layer communication between different layers/planes and should not be confused with the events used by the FSM itself. The events in the FSM are strictly defined by the chosen MAC protocol and dictate the state transitioning process that allows the control plane to manage medium access. The global events such as $SESSION\_ROUTING$ that transition the FSM from an IDLE state are usually set in the decision plane after routing decision has been made. Some other examples of such global events used throughout AXL include $SESSION\_PROCESSING$ event which is used by the session manager to indicate that the node is busy processing a session and $START\_SENSE$ event that signals the PHY layer that it is time to perform spectrum sensing. Therefore, we can state that the overall AXL framework follows an event driven design.

*Register Plane.* The register plane is essentially a node database used to share information across layers/planes. Although the register plane does not perform any computation and does not have any decision making ability, it is an integral part in the overall cross-layer design. The register plane can be considered as a central information hub that can be accessed by different layers/planes of the AXL framework. Data sharing among multiple processes is achieved through Python manager dictionaries. The global information that needs to be shared among all layers is stored in a manager dictionary which allows for only one process to read or write information in the register plane at a time. The main dictionaries that reside in the register plane are a global register dictionary (GRD), global values dictionary (GVD) and session backlog dictionaries (SBD).

AXL nodes learn about their environment by overhearing control packets on the CCC. Each node stores local information in a node dictionary in the GRD. The node dictionary is appended to every control packet sent on the CCC. The information in the node dictionary is continuously updated as new information becomes available. Node dictionary information includes IP and MAC addresses, the node location, local noise plus interference, session packet queue lengths, current routing algorithm among others. Nodes maintain a copy of their own node dictionary, as well as a copy of its neighbor's dictionary in the GRD. The GRD also contains information like the designated frequencies, possible next hops and neighbors. The GVD stores the current routing decision parameters as well as the current state of the FSM. SBD has a list of all local sessions and their most up to date packet queue lengths. The routing algorithm is able to access this information stored in the register plane as it optimizes the routing parameters. Other layers/planes can similarly read or write information in the register plane as needed.

*Physical (PHY) Layer.* The PHY layer is easily separable from the rest of the framework as the goal is to allow the integration of different radio front ends and signal processing software. The PHY layer consists of a hierarchical implementation where the lowest level includes signal processing software specific libraries such as GNU Radio and a universal hardware driver (UHD) interface used with the universal software radio peripheral (USRP) family of products from Ettus. The PHY hierarchical module, consists of functions that are directly accessible by the control layer and the register plane. This implementation allows for a very simple interface between AXL and a PHY layer making this design SDR hardware agnostic.

Fig. 10 depicts the current configuration of a five-node network that is arranged in the form of a grid topology. Each node consists of two USRP N210s (one for control link and the other used as data link) connected to each other via MIMO cable. The SBX and CBX daughterboards are used with the radios, which cover frequency ranges from $400\ \mathrm{MHz}$ to $4.4\ \mathrm{GHz}$ and $1.2\ \mathrm{GHz}$ to $6\ \mathrm{GHz}$ respectively. The receivers (USRP N210s), provide up to $40\ \mathrm{MHz}$ of instantaneous analog bandwidth. The analog-to-digital and digital-to-analog converters on the motherboard use a $100\ \mathrm{MHz}$ master clock and sample at $100\ \mathrm{MS/s}$ and $400\ \mathrm{MS/s}$ respectively. The on-board Xilinx Spartan 3A-DSP 3400 FPGA performs the required digital interpolation or decimation to provide the required sampling rate. The host PC interfaces with the USRP using a Gigabit Ethernet (GigE) connection as shown in Fig. 10. The USRPs are controlled using GNU Radio signal processing modules and a UHD interface. DRS is implemented on the AXL framework and the results are discussed in Section 7.

## 7 Testbed Evaluation

We discuss the experimental evaluation of DRS using the AXL framework implemented on five-nodes USRP testbed

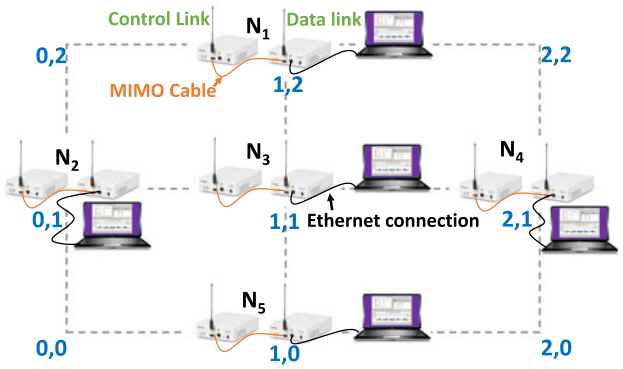


Fig. 10. Layout of the Five-node grid topology.

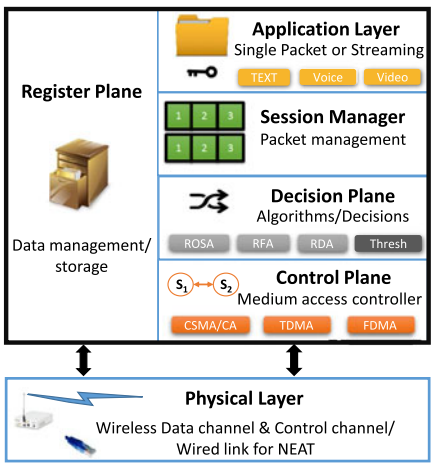


Fig. 11. Adaptive cross-layer (AXL) Framework.

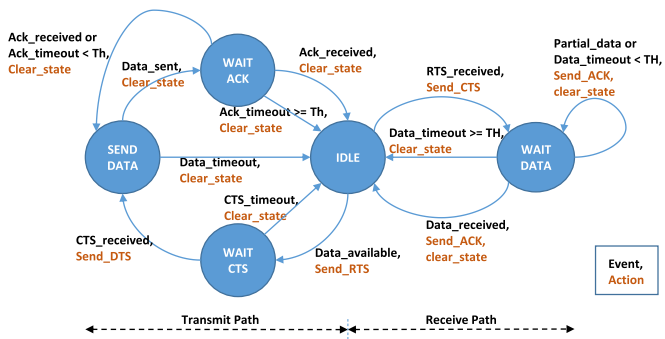


Fig. 12. FSM of MAC protocol.

shown in Fig. 13. In these set of tests, both DRS and ROSA use the same MAC protocol with FSM as shown in Fig. 12. The PHY layer uses Gaussian minimum shift keying (GMSK) implemented using GNU radio with USRPs as the transceivers of the AXL node. The USRP is able to operate at frequency, bandwidth and transmit power level as specified by algorithms (DRS or ROSA) running in the decision plane. This framework provides the required flexibility to implement and evaluate the two cross-layer optimization algorithm (ROSA and DRS). For rapid implementation and feasibility analysis, the AXL framework and the routing algorithms are implemented using Python programming language. The advantage of using the Python is ease of programming and faster development turnaround time. The drawback is large delays incurred by the framework [44], [45]. In future, we plan to move the implementation of the


Fig. 13. Five-node USRP testbed.

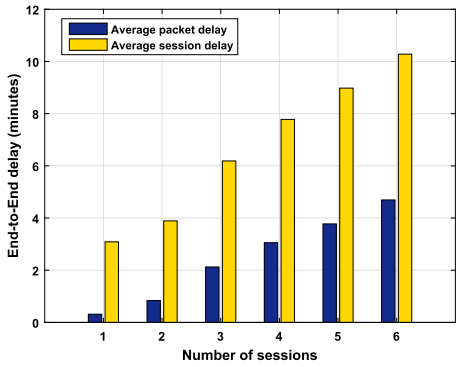


Fig. 14. EED versus No. of Sessions.

TABLE 5
Parameters to Baseline End-to-End Delay

| Parameters | Baseline Test |
|---|---|
| Packet size | 1250 Bytes |
| Number of packets | 3000 |
| Number of sessions | 1 to 6 |
| Source rate | 200 kbits/s |
| Source duration | 150 s |
| Routing algorithm | ROSA |
| No. of seeds | 5 |
| Maximum transmit power | 20 dBm |

AXL framework to the kernel space using C programming language. Therefore, it is important to obtain a baseline for the delay experienced in the network so that we can choose appropriate deadlines for the experimentation process.

### 7.1 Establishing Baseline

In this section, we establish a baseline for the average end-to-end delay (EED) experienced by the SDR based testbed which is the intended platform for evaluating DRS. To accomplish this, ROSA is used as the the default routing algorithm. Accordingly, we calculate the average delay experienced by packets to traverse from source to destination as the number of session increases. The network parameters used for determining the EED experienced by the current configuration of the SDR based ad-hoc network is listed in Table. 5.

Fig. 14 depicts the average EED experienced by each packet and the average EED experienced by the entire session as the number of sessions in the network increase. The EED experienced by the packet is calculated as the duration between the packet generation at the source node and packet arrival at the destination node. Similarly, EED experienced by each session represents the time between the generation of the first packet at the source node and the arrival of the last packet at the destination node. As

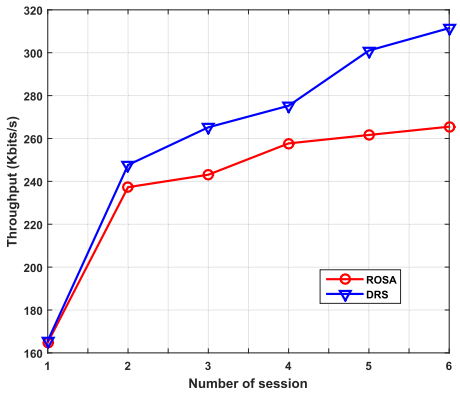


Fig. 15. Test 1: $\eta$ versus No. of sessions.

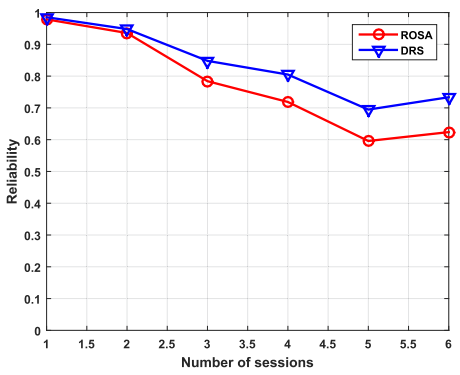


Fig. 16. Test 1: $\rho$ versus No. of sessions.

TABLE 6
Parameters for Testbed Evaluation

| Param. | Test 1 | Test 2 | Test 3 |
|---|---|---|---|
| Deadlines | Odd sess. 3 min<br>Even sess. 15 min | Odd sess. 3 min<br>Even sess. 15 min | 60 min |
| Session start | $t = 0$ s | $t = [0, 2]$ min | $t = 0$ s |
| $\tau$ | $10^{-6}$ | $10^{-6}$ | $10^{-6}$ |
| Seeds | 30 | 30 | 30 |

expected, the average delay increases in both cases with increasing number of sessions. Examining these delay values, we can clearly see the impact of the Python based implementation of the overall framework. Nevertheless, we can use this baseline to choose appropriate deadlines for this network that would enable us to compare the performance of two cross-layered algorithms (DRS and ROSA). Accordingly, for the next set of tests, we choose 3 min as the smaller stringent deadline and 15 min as the larger deadline.

### 7.2 DRS and ROSA

In this section, the effective throughput and reliability of DRS and ROSA are evaluated on the five-node USRP testbed. In the current implementation, the MAC protocol is able to recover any loss of packet that occurs due to channel using retransmission. Therefore, loss of packets only takes place at the destination when the packet reach after the specified deadlines. In addition to parameters listed in Table. 5, this set of experiments use the parameters in Table 6.

In the first set of tests, we used multiple sessions that started at the same time and performance of the SDR based network was evaluated as the number of sessions increased. It is evident from Fig. 15 that for the given network

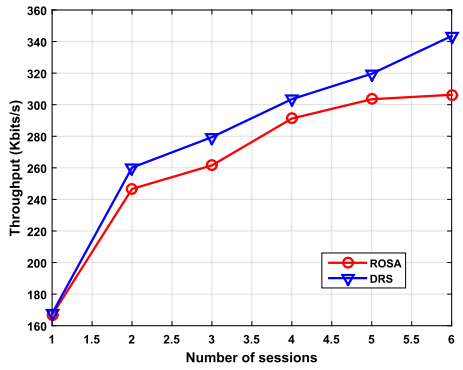


Fig. 17. Test 1: $\eta$ versus No. of sessions.

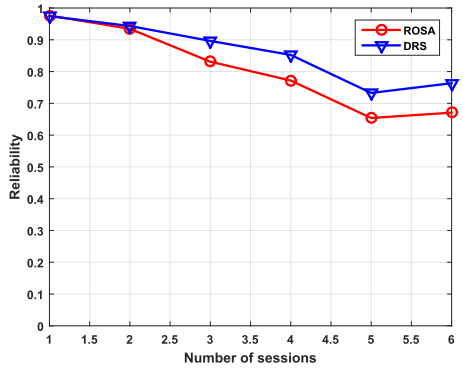


Fig. 18. Test 1: $\rho$ versus No. of sessions.

configurations DRS outperforms ROSA in terms of effective throughput as soon as there are two sessions in the network. The performance trend continues as the number of session increases achieving up to 17 percent improvement over ROSA. This is because DRS is able to manage multiple sessions adaptively to ensure that the effective throughput of the network is maximized. Similar behavior is also observed in Fig. 16 which compares the reliability of DRS and ROSA as the number of packet increases.

In contrast to the first test, the source nodes are set to choose a random time to start the session in the second test. This would imply that different sessions will end at different times leading to the last packet problem in networks using traditional backpressure based algorithm like ROSA. The effective throughput and reliability of Test 2 are depicted in Figs. 17 and 18. As expected DRS outperforms ROSA in terms of effective throughput (up to 12 percent improvement) and reliability (up to 13 percent improvement) even when the the network has finite sessions starting at different times. This is due to the fact that the use of VQL prevents the network from experiencing the last packet problem. VQL keeps increasing with time even if the actual queue length does not change. In the final test, we use a very large deadline (60 min). The goal was to evaluate if there is any degradation in performance of DRS compared to ROSA when the deadlines are large enough to be close to negligible. As shown in Fig. 19, there is no significant loss in performance on using DRS compared to ROSA even in scenarios where the deadlines are long enough to be insignificant. Overall, these tests follow the same trend as simulations discussed in Section 5. The gains observed with experiments is much smaller than with simulations due to the smaller network size. We believe larger benefit can be attained on a larger network deployment. These set of tests

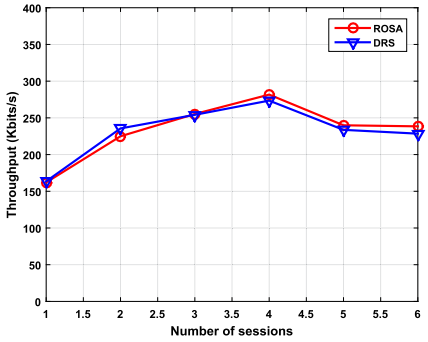


Fig. 19. Test 3: $\eta$ versus No. of sessions.

provide validity to the proposed algorithm to be effective in cognitive network that provides the flexibility to adapt according to the given scenarios.

## 8 Conclusions

We proposed a novel distributed deadline-based joint routing and spectrum allocation algorithm to maximize the effective network throughput. The DRS adapts according to available resources and is capable of handling sessions with different deadline requirements. DRS enables every node in the network to choose optimal session, next hop, frequency and transmit power with an objective to deliver maximum number of packets to their intended destination before the specified deadline. Though DRS is designed for tactical ad-hoc networks, its application can be extended to any wireless ad-hoc network that handles sessions with different QoS based deadline requirements. We have performed extensive simulations to compare the performance of DRS with ROSA and showed up to 35 percent improvement in effective throughput and up to 26 percent improvement in reliability of the network. Furthermore, we have successfully overcome the challenges of implementing the proposed algorithm on a cross-layer framework based software defined testbed. The experiments conducted on the testbed showed DRS outperforming ROSA in terms of effective throughput (up to 17 percent) and reliability (up to 13 percent). This helped us accomplish the secondary objective of this paper which was to validate the flexibility and further advance the COmBAT framework by implementing novel cross-layer DRS algorithm.

## Acknowledgments

(a) Contractor acknowledges the Government's support in the publication of this paper. This material is based upon work supported by the US Air Force Research Laboratory under Award No. FA8750-14-C-0098 and Award No. FA8750-16-C-0086. (b) Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL. A preliminary shorter version of this paper appeared in the *Proceedings of IEEE Globecom 2016* [1].

## References

[1] J. Jagannath, T. Melodia, and A. Drozd, "DRS: Distributed deadline-based joint routing and spectrum allocation for tactical Ad-hoc networks," in *Proc. IEEE Global Commun. Conf.*, Dec. 2016, pp. 1–6.

[2] L. Ding, T. Melodia, S. Batalama, J. Matyjas, and M. Medley, "Cross-layer routing and dynamic spectrum allocation in cognitive radio Ad Hoc networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 4, pp. 1969–1979, May 2010.

[3] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," in *Proc. IEEE Conf. Decision Control*, Dec. 1990, pp. 2130–2132.

[4] V. Brik, E. Rozner, S. Banerjee, and P. Bahl, "DSAP: A protocol for coordinated spectrum access," in *Proc. IEEE Int. Symp. New Frontiers Dynamic Spectr. Access Netw.*, Nov. 2005, pp. 611–614.

[5] M. Bkassiny, S. Jayaweera, Y. Li, and K. Avery, "Optimal and low-complexity algorithms for dynamic spectrum access in centralized cognitive radio networks with fading channels," in *Proc. IEEE Veh. Technol. Conf.*, May 2011, pp. 1–5.

[6] Y. Yuan, P. Bahl, R. Chandra, T. Moscibroda, and Y. Wu, "Allocating dynamic time-spectrum blocks in cognitive radio networks," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Sep. 2007, pp. 130–139.

[7] R. Doost-Mohammady, M. Y. Naderi, and K. R. Chowdhury, "Spectrum allocation and QoS provisioning framework for cognitive radio with heterogeneous service classes," *IEEE Trans. Wireless Commun.*, vol. 13, no. 7, pp. 3938–3950, Jul. 2014.

[8] L. Luo and S. Roy, "Analysis of dynamic spectrum access with heterogeneous networks: Benefits of channel packing scheme," in *Proc. IEEE Global Commun. Conf.*, Nov. 2009, pp. 1–7.

[9] A. Alshamrani, X. S. Shen, and L. L. Xie, "QoS provisioning for heterogeneous services in cooperative cognitive radio networks," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 4, pp. 819–830, Apr. 2011.

[10] A. Dimakis and J. Walrand, "Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits," *Trans. Advances Appl. Probability*, vol. 38, no. 2, pp. 505–521, Jun. 2006.

[11] C. Joo, X. Lin, and N. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multi-hop Wireless networks," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2008, pp. 1103–1111.

[12] M. Leconte, J. Ni, and R. Srikant, "Improved bounds on the throughput efficiency of greedy maximal scheduling in Wireless networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 709–720, Jun. 2011.

[13] R. Laufer, T. Salonidis, H. Lundgren, and P. L. Guyadec, "A cross-layer backpressure architecture for Wireless multihop networks," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 363–376, Apr. 2014.

[14] P. van de Ven, S. Borst, and S. Shneer, "Instability of MaxWeight scheduling algorithms," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2009, pp. 1701–1709.

[15] L. Bui, R. Srikant, and A. Stolyar, "A novel architecture for reduction of delay and queueing structure complexity in the back-pressure algorithm," *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1597–1609, Dec. 2011.

[16] M. Kurth and J.-P. Redlich, "Delay properties of opportunistic back-pressure routing in CSMA-based wireless mesh networks," in *Proc. IEEE Int. Symp. Pers. Indoor Mobile Radio Commun. Workshops*, Sep. 2010, pp. 479–484.

[17] D. Xue and E. Ekici, "Delay-guaranteed cross-layer scheduling in multihop Wireless networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1696–1707, Dec. 2013.

[18] H. Xiong, R. Li, A. Eryilmaz, and E. Ekici, "Delay-aware cross-layer design for network utility maximization in multi-hop networks," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 5, pp. 951–959, May 2011.

[19] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, P. Whiting, and R. Vijayakumar, "Providing quality of service over a shared wireless link," *IEEE Commun. Mag.*, vol. 39, no. 2, pp. 150–154, Feb. 2001.

[20] B. Ji, C. Joo, and N. Shroff, "Delay-based back-pressure scheduling in multihop Wireless networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1539–1552, Oct. 2013.

[21] B. Sadiq and G. de Veciana, "Throughput optimality of delay-driven MaxWeight scheduler for a wireless system with flow dynamics," in *Proc. Annu. Allerton Conf. Commun. Control Comput.*, Sep. 2009, pp. 1097–1102.

[22] S. Liu, E. Ekici, and L. Ying, "Scheduling in multihop Wireless networks without back-pressure," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1477–1488, Oct. 2014.

[23] D. Xue, R. Murawski, and E. Ekici, "Capacity achieving distributed scheduling with finite buffers," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 519–532, Apr. 2015.

[24] B. Ji, C. Joo, and N. Shroff, "Exploring the inefficiency and instability of Back-Pressure algorithms," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2013, pp. 1528–1536.

[25] P. Van de Ven, S. Borst, and L. Ying, "Spatial inefficiency of MaxWeight scheduling," in *Proc. Int. Symp. Model. Optimization Mobile Ad Hoc Wireless Netw.*, May 2011, pp. 62–69.

[26] M. Xiao, J. Wu, H. Huang, L. Huang, and W. Yang, "Deadline-sensitive opportunistic utility-based routing in cyclic mobile social networks," in *Proc. IEEE Int. Conf. Sens., Commun. Netw.*, Jun. 2015, pp. 301–309.
[27] C. Lee and K. I. Kim, "A deadline aware DTN approach based on epidemic routing," in *Proc. IEEE Int. Symp. Netw. Comput. Appl.*, Aug. 2014, pp. 41–44.
[28] Q. Fu, B. Krishnamachari, and L. Zhang, "DAWN: A density adaptive routing for deadline-based data collection in vehicular delay tolerant networks," *Tsinghua Sci. Technol.*, vol. 18, no. 3, pp. 230–241, Jun. 2013.
[29] K. R. Chowdhury and M. D. Felice, "SEARCH: A routing protocol for mobile cognitive radio ad-Hoc networks," in *Proc. IEEE Sarnoff Symp.*, Mar. 2009, pp. 1–6.
[30] L. Ding, K. Gao, T. Melodia, S. Batalama, D. Pados, and J. Matyjas, "All-spectrum cognitive networking through jointly optimal distributed channelization and routing," *IEEE Trans. Wireless Commun.*, vol. 12, no. 11, pp. 5394–5405, Nov. 2013.
[31] V. Srivastava and M. Motani, "Cross-layer design: A survey and the road ahead," *IEEE Comm. Mag.*, vol. 43, no. 12, pp. 112–119, Dec. 2005.
[32] L. D. Mendes and J. J. Rodrigues, "A survey on cross-layer solutions for wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 34, no. 2, pp. 523–534, 2011.
[33] B. Fu, Y. Xiao, H. J. Deng, and H. Zeng, "A survey of cross-layer designs in wireless networks," *IEEE Commun. Surveys Tutorials*, vol. 16, no. 1, pp. 110–126, Jan-Mar. 2014.
[34] I. Al-Anbagi, M. Erol-Kantarci, and H. T. Mouftah, "A survey on cross-layer quality-of-service approaches in wsns for delay and reliability-aware applications," *IEEE Commun. Surveys Tutorials*, vol. 18, no. 1, pp. 525–552, Jan.-Mar. 2016.
[35] R. Subramanian, B. Drozdenko, E. Doyle, R. Ahmed, M. Leeser, and K. R. Chowdhury, "High-level system design of IEEE 802.11b Standard-Compliant Link Layer for MATLAB-Based SDR," *IEEE Access J.*, vol. 4, pp. 1494–1509, Apr. 2016.
[36] G. Sklivanitis, E. Demirors, A. Gannon, S. N. Batalama, D. A. Pados, and S. N. Batalama, "All-spectrum cognitive channelization around narrowband and wideband primary stations," in *Proc. IEEE Global Commun. Conf.*, Dec. 2015, pp. 1–7.
[37] A. L. Drozd, T. Arcuri, J. Jagannath, D. A. Pados, T. Melodia, E. Demirors, and G. Sklivanitis, "Network throughput improvement in cognitive networks by joint optimization of spectrum allocation and cross-layer routing," in *Proc. NATO Symp. Cognitive Radio Future Netw.*, May 2014.
[38] E. Demirors, G. Sklivanitis, T. Melodia, and S. N. Batalama, "RcUBe: Real-time reconfigurable radio framework with self-optimization capabilitites," in *Proc. IEEE Int. Conf. Sens. Commun. Netw.*, Jun. 2015, pp. 28–36.
[39] J. Jagannath, H. Saarinen, W. Timothy, J. O'Brien, S. Furman, T. Melodia, and A. Drozd, "COmBAT: Cross-layer based testbed with analysis tool implemented using software defined radios," in *Proc. IEEE Conf. Military Comm.*, Nov. 2016, pp. 1261–1266.
[40] M. C. Ng, K. E. Fleming, M. Vutukuru, S. Gross, Arvind, and H. Balakrishnan, "Airblue: A system for cross-layer Wireless protocol development," in *Proc. ACM/IEEE Symp. Architectures Netw. Commun. Syst.*, 2010, pp. 1–11.
[41] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste, "Enabling MAC protocol implementations on software-defined radios," in *Proc. USENIX Symp. Net. Syst. Des. Implementation*, 2009, pp. 91–105.
[42] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, and F. Gringoli, "Wireless MAC processors: Programming MAC protocols on commodity Hardware," in *Proc. IEEE Conf. Comput. Commun.*, Mar. 2012, pp. 1269–1277.
[43] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *ACM Trans. SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.
[44] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste, "Enabling MAC protocol implementations on software-defined radios," in *Proc. USENIX Symp. Networked Syst. Des. Implementation*, Apr. 2009, pp. 91–105.
[45] K. R. Chowdhury and T. Melodia, "Platforms and testbeds for experimental evaluation of cognitive ad hoc networks," *IEEE Commun. Mag.*, vol. 48, no. 9, pp. 96–104, Sep. 2010.

**Jithin Jagannath** received the master of science degree in electrical engineering from the University at Buffalo, The State University of New York, in 2013. He is working toward the PhD degree in the Department of Electrical and Computer Engineering, Northeastern University. He is currently conducting research with the Wireless Networks and Embedded Systems Laboratory under the guidance of Prof. Tommaso Melodia. He has experience developing SDR based prototypes which includes designing, implementing, and testing different communication protocols and optimized algorithms on hardware for wireless sensor networks. He is also a Sr. associate scientist/engineer at ANDRO Computational Solutions in Rome, NY. He is currently PI of a DHS SBIR effort that aims to develop a novel interference detection device for first responders. He has also been the technical lead in multiple SBIR/STTR efforts including cross-layer networking technology, multi-sensor automatic modulation classification, cyber security, and compressive sensing. His research interest includes software defined radio, visible light communication, cognitive networks, underwater sensor network, and automatic modulation classification.

**Sean Furman** received the BSECE degree from the State University of New York Polytechnic Institute, in Utica, New York, in 2017. In the 2016-2017 academic year, he worked on VANETTA (Vehicular Ad-Hoc Network for Transportation Automation) as his senior project. The project involved design and implementation of vehicular platooning protocols on scaled remote control cars. He joined ANDRO Computational Solutions in 2015 and has contributed to projects involving software defined radio networks. His research interests include wireless communication, automatic control systems, and computing.

**Tommaso Melodia** received the PhD degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, in 2007. He is an associate professor with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA. His research has been supported by the National Science Foundation, Air Force Research Laboratory, and the Office of Naval Research, among others. His current research interests include modeling, optimization, and experimental evaluation of networked communication systems, with applications to ultrasonic intra-body networks, cognitive and cooperative networks, multimedia sensor networks, and underwater networks. He was a recipient of the National Science Foundation CAREER Award and coauthored a paper that was recognized as the ISI Fast Breaking Paper in the field of computer science for February 2009. He was the technical program committee vice chair for IEEE GLOBECOM 2013 and the technical program committee vice chair for Information Systems for IEEE INFOCOM 2013. He serves on the editorial boards of the *IEEE Transactions on Mobile Computing*, the *IEEE Transactions on Wireless Communications*, the *IEEE Transactions on Multimedia*, and *Computer Networks*.

**Andrew Drozd** is the president and chief scientist of ANDRO Computational Solutions in Rome, NY. His fields of expertise include wireless communication, dynamic spectrum access, computational electromagnetics (CEM), development of electromagnetic environmental effects (E3) tools for cosite analysis, spectrum exploitation, and interference rejection technologies. He was president of the IEEE EMC Society (2006-2007), a past member of the EMC-S Board of Directors (1998-2008), and is an IEEE fellow. He was also on the Board of Directors of the Applied Computational Electromagnetics Society (ACES) (2004-2010). He is an iNARTE certified EMC Engineer and has authored more than 160 technical papers, reports, and journal articles.