Wireless Netw (2012) 18:257–275
DOI 10.1007/s11276-011-0398-8

Application-aware dynamic spectrum access

Ashwini Kumar · Kang G. Shin

Published online: 10 November 2011

Abstract Dynamic Spectrum Access (DSA) allows unlicensed wireless devices to opportunistically access unoccupied licensed spectrum bands. DSA yields efficient spectrum utilization which can greatly improve wireless networking performance. In this paper, we advocate application-awareness to effectively manage the side-effects of DSA that can offset its benefits by adversely impacting application QoS. Simple application hints are found to be able to serve as key inputs in evaluating current spectrum conditions relative to application needs, leading to an informed DSA mechanism that minimizes the impact of undesirable DSA side-effects. Towards this goal, we propose a wireless service architecture called Context-Aware Spectrum Agility (CASA). The key elements of CASA are: (a) semantic dependency equations that provide the relationship between application-layer QoS state and lower-layer DSA parameters, (b) CASA Algorithm that adapts DSA parameters and activities to better suit application needs, and, (c) a low overhead interface to provide application context to DSA. CASA has been explicitly designed with the goals of practical deployment, low overhead operation, and is compatible with any DSA protocol. Compared to state-of-art DSA, the deployment of CASA along with DSA protocols is shown to improve QoS metrics, such as delay and jitter, by an average of 30 and 64%, respectively. CASA is also found to match the application QoS demands for more than 90% of the duration of a communication session—a 300+% improvement over conventional application-agnostic DSA.

Keywords Dynamic Spectrum Access (DSA) · Wireless networks · QoS · Software-Defined Radio (SDR) · Cross-layer · Application hints

1 Introduction

Dynamic Spectrum Access (DSA), *Spectrum Agility* (SA), or *White Space Networking* [2] is a new wireless networking paradigm that aims to solve the spectrum scarcity problem in wireless communications. DSA relies on opportunistic exploitation of licensed channels by unlicensed devices (also called *secondary users* or SUs). Such unlicensed accesses must occur when authorized licensee devices (also called *primary users* or PUs) are not concurrently accessing the channel, i.e., during *spectrum white spaces*. Recent surveys [3, 4] have shown existence of abundant spectrum spaces in the licensed spectrum. The potential benefits of spectrum-agile operations has led regulatory bodies, like FCC, to move towards opening licensed channels for DSA [5, 6]. A growing interest is being witnessed in developing DSA-based products, especially for TV bands [7–9].

DSA is still in an early stage of development, and has a much broader scope than TV bands. Fundamentally, DSA is not limited to a particular spectrum region, and can involve opportunistic switching of channels between different spectrum regions. We take this general view of DSA here, in which, apart from switching channels, access mechanisms (or the MAC-PHY protocols) may also need

Preliminary work appeared as a 4-page paper in the proceedings of the ACM MobiCom 2007 [1].

A. Kumar (✉) · K. G. Shin
Real-Time Computing Laboratory, EECS Department,
The University of Michigan, Ann Arbor,
MI 48109-2121, USA
e-mail: ashwinik@eecs.umich.edu

K. G. Shin
e-mail: kgshin@eecs.umich.edu

to be dynamically changed in order to accommodate different wireless characteristics of various spectrum bands. The FCC's establishment of rules that open up TV bands for DSA [5] has stimulated efforts in advancing technologies for DSA across other licensed bands [10, 11].

Gap Between State-of-art DSA and Application QoS: The main motivation behind this paper is our investigation which reveals that though state-of-art DSA leads to gain in spectral resource at the channel level (i.e., the link layer bandwidth) it does not translate into corresponding gain in application performance. DSA's negative side-effects, which are undesirable for network applications, constitute the main cause behind this phenomena. Fundamental operations involved in DSA, like spectrum sensing or channel switches, could cause delays and disruptions to the applications—thereby introducing QoS degradation. Containing the interference to incumbents is the key requirement for opportunistic usage of licensed spectrum. Thus, any PU activity adds to the interruptions suffered by SU applications. Further, DSA may also result in link capacity fluctuations due to a reduction in frequency-width or less-efficient MAC-PHY schemes on a new channel. In the worst case, session handovers, terminations and re-establishments may occur, exacerbating application QoS degradation.

Therefore, contrary to the current perception in the DSA research community, we argue that achieving gains in link layer capacity is not good enough at the application level. We make two mutually contradictory observations about state-of-art DSA that define the main theme of this paper.

O1. DSA aims to improve network application performance.

O2. State-of-art DSA is agnostic to application needs.

Clearly, O1 is orthogonal to O2. Hence, DSA incurs unwarranted side-effects, directly impacting application QoS. We postulate that there exists a tension between DSA operations and a network application's traffic requirements, which necessitates cooperation between the two entities to effectively achieve DSA's goals.

While there have been recent work in accounting for QoS in DSA [10, 12, 13], only low level QoS metrics have been considered, e.g., link SNR and BER. Thus, they are unable to accurately capture specific application-layer QoS demands which have high-level semantics.

Proposed Approach: In this paper, we propose an adaptive application-aware service framework for DSA in order to improve application performance. We call this DSA-enhancement *Context-Aware Spectrum Agility* (CASA), where *context* comprises application QoS hints as well as current spectrum conditions. The high level application context is first processed through *semantic matching* to determine their dependency on low level DSA parameters. CASA exploits the short-term correlation of the recent networking state with near future in making DSA application-aware. The adaptivity scheme of CASA is built-upon reward-based Reinforcement Learning [14] methods. The main goal of CASA is to minimize the undesirable impacts of DSA, and thus, re-enforce and enhance the merits of DSA as a performance-improving feature in wireless devices. Our evaluation shows that CASA deployment improves application performance significantly during DSA operation, and indirectly, makes DSA resilient under stringent application QoS requirements.

Our solution is inspired by the advantages shown in the areas of power management and wireless network selection [15, 16], when application-awareness is incorporated in corresponding optimization algorithms. These works highlight that application behavior/demands are the most significant aspect of networking context, and their effective integration with networking protocols can be instrumental towards improving application performance.

Contributions: The main contributions of this paper are:

(1) Identification of the harmful side-effects in state-of-art DSA that adversely impact applications;
(2) Introduction of the concept of semantic matching between application state and lower-layer parameters via the development of semantic dependency equations;
(3) Development of a low overhead, easily-deployed service architecture (i.e., CASA) to make DSA application-aware and mitigate its side-effects; and,
(4) Prototype implementation and comprehensive testbed evaluation of CASA.

Organization: The paper is organized as follows. We will begin with a description of the system model. Section 3 presents the necessary background and motivation. Sections 4–8 detail the design of CASA, while Sect. 9 describes its implementation. Section 10 evaluates CASA, and Sect. 11 discusses prior related work. Section 12 concludes this paper.

2 System model

We consider DSA in the secondary wireless service market model, as described below. The service structure for the secondary market is still evolving, but is expected to closely resemble the currently existing consumer wireless service model (e.g., cellular service), as their service infrastructure already exists with proven effectiveness and success.

2.1 Principal model

Device: The devices in our system model are general-purpose computing units with wireless networking

capability (e.g., laptops and smartphones)—each equipped with one DSA-capable wireless interface. The wireless interface is composed of a highly-reconfigurable Software Defined Radio (SDR) or cognitive radio (CR) together with a highly-tunable antenna [17– 20]. A basic prototype of a CR (though with limited capabilities) is the software defined radio USRP2 [21].

We consider a single SDR-based data interface scenario here because of its cost-size-design advantage and simplicity of analysis. However, our proposed solution also applies to multi-interface (or multi-SDR) devices.

To maintain the generality and compatibility of the proposed solution, we do not assume any restrictions on the actual DSA protocol. Thus, for example, the SU may utilize the same interface for spectrum sensing or have a separate specialized *spectrum-sensor* hardware to minimize interruptions to data-transfers during sensing. External sensing infrastructure (e.g., sensor network or sensing server) can also be used for collecting spectrum-condition information.

Network: We consider infrastructure networks similar to typical wireless service provider networks for our system model. Each SU, by virtue of its DSA capability, can migrate to a suitable licensed channel where it can connect to the secondary gateway/base-station of such networks. Thus, the SU is a client in such first/last-mile wireless access networks, as shown in Fig. 1. The SU employs DSA to select channels flexibly across the wireless spectrum in order to avail of network services.

Example: A typical scenario would consist of a PDA-like convergent DSA-capable device, which is in the range of multiple edge access networks on different channels, possibly of different types (e.g., WiMAX, cellular, 802.22, etc). One or more of the networks can be *primary* to the device (i.e., authorized to use it at any time), while other networks are available on a secondary basis and can be accessed opportunistically.

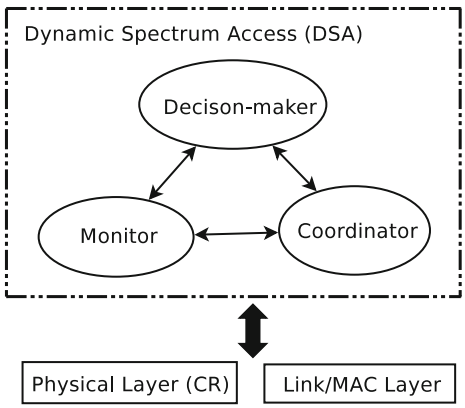


Fig. 2 Functional model of a typical DSA protocol

2.2 Other models

As clear from the prior discussion, we focus primarily on a device-centric model, e.g., clients in edge access networks. For completeness sake, we also mention below other types of system models where our proposed solution CASA can be incorporated with minor changes.

CASA can be applied to generic one-hop point-to-multipoint DSA networks, such as WRANs in the IEEE 802.22 standard [22]. However, in these cases, network-wide DSA decisions need to be coordinated properly in order to avoid conflicts. Effective DSA coordination together with a central decision-making entity (e.g., a base station) in these type of networks can avoid any spectrum management conflicts.

Further, mobile devices in a fully ad-hoc/distributed DSA network can also deploy the proposed CASA service architecture to optimize local DSA performance. However, coordination of CASA-related actions in the network introduces additional overhead and may require complex modifications to the existing DSA protocols. We leave this as our future work.

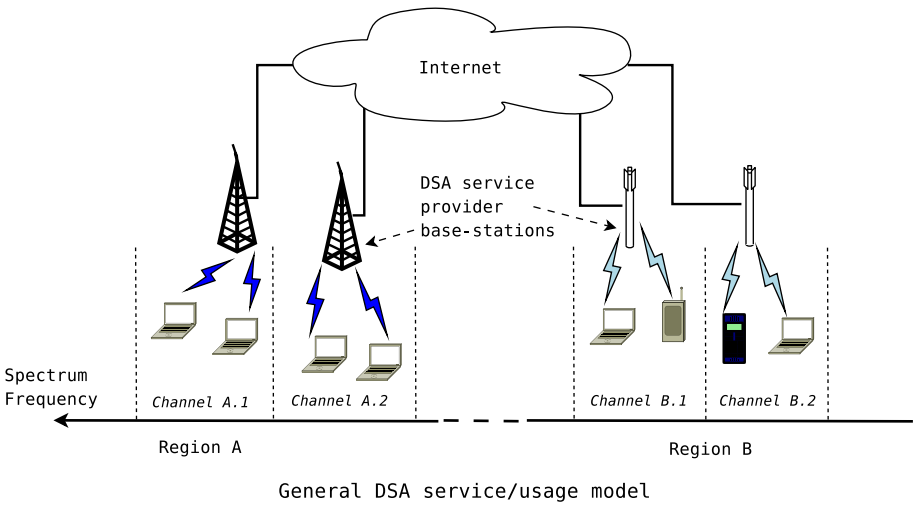


Fig. 1 Client devices in first- or last-mile wireless access networks constitute the system model

2.3 Assumptions

We make two minor and non-restrictive assumptions during development of CASA.

- The average raw physical layer data-rate and the application layer efficiency for each of the MAC-PHY scheme to be deployed on a channel is assumed to be known.
- The average utilization fraction of the channels, including their primary and secondary user utilization components, are known. These values are typically available from the underlying DSA protocol.
- The packet arrival rate at the link layer during very short durations is assumed to be constant. This assumption is especially valid for many QoS-sensitive traffic like Internet multimedia streaming and VoIP [23].

3 Background and motivation

3.1 Background

To better understand the issues of focus in this work, we first provide a brief overview of contemporary (or state-of-art) DSA architecture.

DSA, as a networking module, spans physical and link layers. The physical layer aspect is captured by SDRs/CRs which provide the radio capability necessary for DSA. More relevant to this paper are the higher-layer MAC aspects that manage how DSA operates. There have been numerous DSA protocol proposals in literature [24–27]. The process for DSA standardization has also begun [22, 28]. However, at the time of writing this paper, there is no consensus in the research community on a standard DSA protocol. Thus, instead of selecting one protocol proposal and disregarding others, we consider the state-of-art DSA from a *functional abstraction* viewpoint.

Our study of several DSA protocols reveals that state-of-art DSA protocols share certain key functions that must be performed to achieve DSA (Fig. 2). This observation enables us to create an abstract model of DSA which forms the basis of our solution development. Our approach ensures the generality of our analysis and proposed methods, despite the evolving nature of current DSA research. It permits as to study and understand the behavior of fundamental DSA components, without incurring incompatibility problems or selectivity bias of choosing one (or a few) specific DSA protocols.

From a functional abstraction perspective, DSA consists of three main components: spectrum sensing, spectrum-use decision making, and coordination (see Fig. 5).[1]

[1] These functions are realized through the underlying MAC (e.g., for coordination) and PHY schemes (e.g., for sensing).

The *spectrum sensing* component scans channels in the spectrum and acquires relevant time-variant characteristics for each channel. A DSA protocol typically maintains a list of channels together with their average spectrum white-spaces. This list is referred to as the *Spectrum Opportunity Map* (SOM) [29].

The *spectrum-use decision making* component determines the channel for secondary devices to use, and invokes the channel-switching and coordination procedure, if needed. For this, it analyzes the information gathered by the sensing component. For instance, it is invoked when an incumbent signal is detected on the current channel.

The *coordination* component orchestrates DSA decisions in a multi-node DSA network. For instance, the coordination component ensures that the SUs are on the same channel, thus maintaining their inter-communication. Many of the proposed DSA protocols use control channels to exchange control information in order to accomplish this coordination.

As seen from its abstract function model, state-of-art DSA's main design objective is to gain as much spectral resources as possible. Channel utilization is the the main decision metric used for this purpose. Gain in spectral resource is expected to translate into additional link layer bandwidth, thus resulting in performance advantage of DSA. For more details on the design and techniques involved in the aforementioned DSA components, interested readers are pointed to the references described in Sect. 11.

3.2 Motivation

As seen in Sect. 3.1, there are various functionally-disjoint components in DSA. Therefore, their overhead and interplay may have unwarranted side-effects on end-user applications in typical networking environments. This observation motivated us to investigate the behavior of DSA in direct relation to application QoS requirements. Based on our analysis, we confirmed the existence of the following problem with state-of-art DSA: *Despite resulting in gain of spectral resources, DSA produces unwanted impact on application QoS performance due to disruptive side-effects produced by its basic functions*. We next elaborate on the main causes for the observed problem, and also present experimental proof confirming the same. While [30, 31] also discuss some of these DSA's side-effects, their investigation is limited to transport layer rather than application QoS which is of focus here.

Impact of spectrum sensing: Effective spectrum sensing requires scheduling of *quiet periods* (QPs), during which no data can be transmitted/received. Depending on the channel characteristics and the sensing scheme used, each QP may vary from tens to hundreds of milliseconds.

For example, a QP in IEEE 802.22 [22] may last for 25 ms. Further, a QP may be scheduled very often, as frequent sensing improves DSA performance. Frequent spectrum sensing is also required to comply with regulatory guidelines on quick incumbent detection [5]. FCC requires incumbent signal to be detected within 2 s in TV bands [5].

Thus, the overhead of sensing (due to QPs) can adversely impact application QoS by introducing extra delay & jitter, and reducing usable bandwidth in the process (see Fig. 3). The negative side-effects of sensing are magnified when the channel is narrowband and offers lower capacity (e.g., TV bands are 6 MHz wide). Sometimes, due to size/cost/hardware constraints, a separate spectrum-sensor may not be available. In such cases, "out-of-band" sensing will add to "in-band" sensing impact by significantly multiplying the amount of QPs scheduled on the only available wireless interface. External sensing approaches (e.g., sensing database) may mitigate the sensing impact at the cost of additional infrastructure. However, this does not fully eliminate sensing overhead as in-device sensing would still be required for validation purpose and to detect any unexpected incumbent activity, or when external sensing data is unavailable.

Some prior works like [32–35] attempt to reduce the impact of spectrum sensing by estimating the channel model, or by incorporating sensing history in predicting the future state of the channels. However, these methods are one-dimensional as they do not exploit their optimization strategy in relation to the actual application QoS demands in the system. This limits the effectiveness of such approaches.

Impact of switching channels: Channel-switching is fundamental to DSA. Like spectrum sensing, it has a similar disruptive side-effect. A channel-switch interrupts application-traffic for durations lasting several milliseconds. This includes the time to reset the wireless interface(s), and more importantly, loading MAC-PHY protocols to access the new channel and complete management tasks like association or authentication with the secondary base-station.

Channel-switching may not happen as frequently as spectrum sensing, and hence, its delay/jitter impact is lower. However, it can produce the adverse side-effect of reducing the available bandwidth for application traffic. For example, the new channel may have a lower frequency-width (narrowband channel) than the previous channel, which could result in a sudden decrease in link capacity. Capacity reduction can also be contributed by the usage of a comparatively less-efficient MAC-PHY schemes in the new channel, even though the channel is better in terms of utilization and radio characteristics.

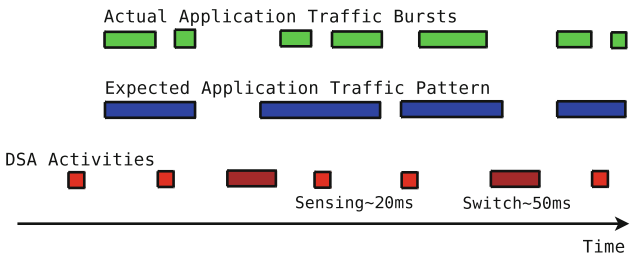


Fig. 3 Disruptions caused by DSA operations result in a fractured flow of application traffic

Certain approaches, like [28, 36], advocate the maintenance of *Backup Channels* to reduce the overhead of channel switching. While useful to a certain extent, the channels are monitored and picked in an application-agnostic fashion—based only on the wireless channel conditions.

Impact of incumbent protection: Very limited interference to the PUs is of critical importance to DSA (ideally there should be no interference to incumbents). Hence, a SU must stop its transmission or switch channels, whenever it detects PU activity. If incumbent activity is prolonged, then SU application traffic is effectively stopped, resulting in severe application QoS degradation. Even short-term incumbent activity can adversely affect QoS-sensitive applications if they occur frequently. Thus, average incumbent utilization metric may not be sufficient to determine the quality of a channel with respect to application requirements—information about incumbent access pattern must be taken into account.

Other unwanted QoS impacts of DSA: Additional delays can also be introduced due to coordination of devices in a multi-SU or adhoc-type networks. Though not the primary focus of this work, many contemporary DSA protocols require periodic listening to a *control* channel for coordination [2], which can be a significant disruption to application-traffic in single-interface systems.

Compounding impact of side-effects: The decline of service quality because of the negative side-effects of DSA usually turns out to be more significant and lasts for a much longer duration, because other layers in the protocol stack may perceive incorrect network conditions. For example, TCP may view the sensing-introduced delays as congestion in the network. Similarly, QoS-centric protocols like RTP [37] will experience frequent short-term adjustment periods. Further, application sessions may be completely disrupted, leading to additional overheads of their re-establishment.

Experimental demonstration of the problem: We conducted simple testbed experiments to study the impact of DSA side-effects on application traffic. A laptop equipped with an Atheros wireless interface emulates a SU device that accesses network services through secondary wireless access-points in different channels. To emulate state-of-art DSA, we implemented its function model

(as described in Sect. 3) in the 802.11 MadWifi driver [38]. We use 802.11a as the base MAC-PHY because of its research accessibility in designing experiments to demonstrate the DSA issues. Note that our goal is not to develop any new DSA protocol, but to implement its abstract function model. More implementation details for our testbed can be found in Sect. 9.

The experiment setup consists of three channels with average incumbent utilization of 0.7, 0.5 and 0.3, respectively. Channels 1 and 2 can support the highest 802.11a raw capacity (54 Mbps), while channel 3 is constrained to 30% of the maximum possible value (e.g., TV channels may be freer than ISM channels, but have low capacity due to narrow spectrum-width). The sole secondary user communicates with a fixed host on the Internet and bootstraps on channel 1. We keep the application bandwidth demand at 10 Mbps, which is enough to support multiple high-quality QoS streams (e.g., each G.711 audio + H.261 video typically requires 460 kbps [23]) along with other types of network traffic.

Figure 4 shows the typical behavior of state-of-art DSA in a sample experiment run. The impact of DSA on application traffic in terms of three QoS parameters—throughput, end-to-end (roundtrip) delay, and jitter—are plotted. The graphs show their average values over each second through a 30-second period of the communication session. As seen, the throughput target is not met: in fact, it drops significantly at around 9 s. As channel utilization is the state-of-art DSA's key decision parameter, the SU switches to channel 3 as it has the lowest utilization. However, it fails to take into account its lower capacity relative to the application bandwidth requirements (which is 10 Mbps). This scenario may occur quite often when SUs switch to channels/protocols in different spectrum regions. For example, from a highly utilized unlicensed 2.4 GHz channel to less utilized (but lower capacity) 4G uplink cellular channel opened by its operator for unlicensed operation.

Though the delay and jitter are found to be acceptable on average, they are found to fluctuate significantly on a short time-scale. In particular, jitter is as high as 10 ms in many cases.

Since DSA is perceived as a performance-enhancing benefit, any degradation of service because of introducing DSA would not be attractive, especially for those applications that require certain minimum bandwidth, delay, and jitter guarantees. Therefore, gain in spectral resources must be effectively passed on to the applications.

4 Context-Aware Spectrum Agility (CASA) overview

We propose a novel wireless networking service architecture, called *Context-Aware Spectrum Agility* (CASA), which augments state-of-art DSA by making it application-aware in order to address the issues identified in Sect. 3.2. CASA introduces application-awareness to DSA through the *CASA Algorithm* which takes context information (both application and lower-layer context) as input, and updates key DSA parameters as necessary. CASA Algorithm executes periodically as part of the *Controller* as shown in Fig. 5. This period is referred to as *CASA Epoch* (τ_{casa}). CASA also provides a low overhead interface called *Cross-Layer Interaction Framework (CLIF)* to enable applications to export their QoS context information.

CASA is designed according to the following principles.

- **Be aware of the operational context**. CASA provides the mechanism to provide application hints, while also collecting the lower-layer information.

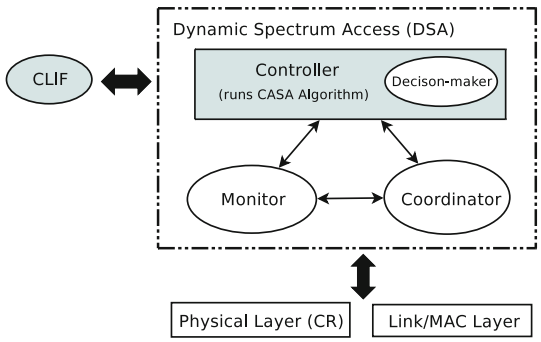
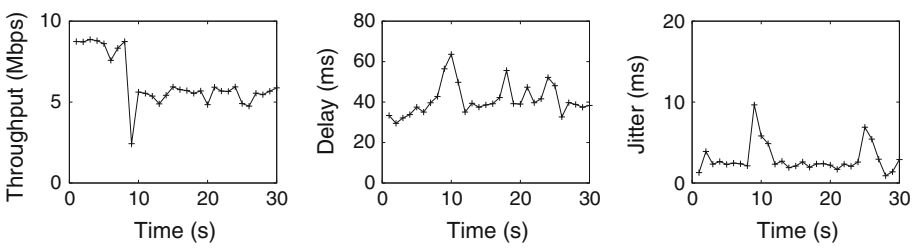


Fig. 5 CASA's main component is the CASA Algorithm, that augments the "Decision-Maker" to form "Controller" in a typical DSA design model, as shown



(a) Throughput **(b)** Delay **(c)** Jitter

Fig. 4 Variation of key QoS metrics with time using state-of-art DSA

- **Be adaptive to changing operating environment**. CASA strives to match application requirements by adapting DSA operations through the CASA Algorithm.
- **Be practical, generic, and ensure minimal overhead**. CASA utilizes well-known low overhead techniques (e.g., exponential averaging) in its design, and can be applied with any DSA protocol.
- **Utilize correlation of recent past with near future**. This is a simple but highly effective principle that has been utilized successfully in many areas of computing, including OS (e.g., CPU scheduling), and networking (e.g., TCP congestion/flow control). CASA uses this principle to guide DSA behavior in immediate future by using recent context information.

Details on various elements involved in CASA are provided in oncoming sections.

5 Context information

5.1 Application context

The application context comprises conventional application QoS attributes. Specifically, they are thresholds for three important QoS metrics for a network application—*minimum bandwidth*, *maximum delay*, and *maximum jitter*. These are considered on an end-to-end and per-session basis from application layer perspective. The "bandwidth" here implies the average end-to-end throughput for an application. Similarly, "delay" implies the average end-to-end latency of the packets in the communication stream, while "jitter" is the variability of the delay parameter. If t_1, t_2, and t_3 represent the time at which 3 successive packets are received by an application, then the jitter is $|(t_3 - t_2) - (t_2 - t_1)|$ (MPPDV method [37]).

Bandwidth, delay, and jitter are specifically chosen because they are directly impacted by DSA side-effects (as seen in Sect. 3.2). This set of QoS parameters provides a simple but effective abstraction of application-layer QoS demands. These fundamental application QoS parameters capture all of the qualitative QoS phenomena observed for any application from an end-user perspective [23]. For instance, voice/video distortion that is sometimes observed during a video conference session, is the direct outcome of the associated delay and jitter

Currently, CASA does not include the packet loss metric in its set of input application QoS parameters. It relies on existing error detection and retransmission features in the networking stack (e.g., TCP/link retransmissions) to take care of any packet loss along the end-to-end path. From an application QoS perspective, the effect of packet error/loss is primarily reflected in the end-to-end delay and jitter parameters, as well as bandwidth, which is included in the current scheme.

Since most modern QoS-sensitive network applications already monitor and manage QoS metrics, application QoS hints can be provided without significant modifications or additional overhead. For example, multimedia streaming applications typically use RTP [37], which monitors traffic characteristics including QoS information through its RTCP component. Thus, RTP/RTCP can be easily leveraged to provide the necessary application context.

We formalize the application context as follows. Assume that there are n network applications running on a SU device. For every ongoing communication session, application i exports the required bandwidth (b^i_{req}), end-to-end delay (d^i_{req}), and jitter (j^i_{req}). Here we show the analysis for one communication session per-application in interest of presentation clarity. However, CASA supports any number of sessions for each application.

Similar to the "requirement parameters", the applications also provide the current "observation parameters"—bandwidth (b^i_{obs}), delay (d^i_{obs}), and jitter (j^i_{obs}), for the corresponding communication sessions.

Individual applications contexts are combined to generate the following cumulative application context.

(1) $B^{app}_{req} = \sum_{i=1}^{n} b^i_{req}$,
(2) $D^{app}_{req} = \min \{d^i_{req}\}, 1 \le i \le n$,
(3) $J^{app}_{req} = \min \{j^i_{req}\}, 1 \le i \le n$.
(4) $B^{app}_{obs} = \sum_{i=1}^{n} b^i_{obs}$,
(5) $D^{app}_{obs} = \max \{d^i_{obs}\}, 1 \le i \le n$,
(6) $J^{app}_{obs} = \max \{j^i_{obs}\}, 1 \le i \le n$.

Thus, the complete application-layer context is the pairs: (B^{app}_{req}, B^{app}_{obs}), (D^{app}_{req}, D^{app}_{obs}), and (J^{app}_{req}, J^{app}_{obs}).

Aggregation of all application QoS hints in thus-defined manner simplifies CASA design, allowing it to manage diverse set of requirments under a single structure. Thus, it avoids high complexity overhead and scalability issues associated with servicing individual requirements. At the same time, this method guarantees that fulfillment of any aggregated requirement parameter corresponds to the fulfillment of the corresponding individual application QoS requirement, with a very high likelihood. Analogous reasoning can be easily applied for "observation parameters".

In the CASA architecture, applications can update and export their QoS requirements/observation pairs whenever they change. Providing all three of the parameter pairs is optional—an application may also provide a subset of these parameter pairs (or none of them) depending upon its operational needs.

It must be pointed out that there is a subtle but important difference between the QoS abstractions corresponding to

Table 1 List of symbols

Symbol	Description
λ_{pkt}	Rate of app. packets generation
B_{req}^{app}, D_{req}^{app}, J_{req}^{app}	App. layer bandwidth, e2e delay & jitter reqs.
C	Set of channels in the spectrum
c	A channel, $c \in C$
$u(c)$	Total utilization of c
$u^{pu}(c)$, $u^{su}(c)$	Utilization by PUs and SUs, respectively in c
$T_{on}^{pu}(c)$, $T_{off}^{pu}(c)$	Random var. for PU's ON/OFF durations in c
$T_{on}^{su}(c)$, $T_{off}^{su}(c)$	Random var. for SU's ON/OFF durations in c
$t_{sense}(c)$, $r_{sense}(c)$	Duration and rate of sensing in c
$t_{switch}(c1,c2)$	Overall time to switch from channel $c1$ to $c2$
$B(c)$	App. bandwidth estimate for channel c
$D(c)$, $J(c)$	Additional delay and jitter when using c
$e(c)$	Efficiency of MAC-PHY protocols used on c
τ_{casa}	CASA Epoch duration
P_{curr}, P_{past}	Current and past push factors
P_{MAX}	Normalization constant for push factor
N	History window (no. of past epochs)

delay/jitter (D_{req}^{app}, D_{obs}^{app}, J_{req}^{app}, J_{obs}^{app}), and those corresponding to bandwidth (B_{req}^{app}, B_{obs}^{app}). Delay or jitter is *transit-additive* in nature—individual network segments traversed along the communication path contribute to produce the cumulative end-to-end delay/jitter as seen by applications. Thus, each link or network segment along the communication path must minimize its individual contribution to these metrics for better application QoS. On the other hand, bandwidth is *transit-reductive* in nature—the overall bandwidth experienced by the application is the least bandwidth experienced along the entire communication path. Therefore, bandwidth provided by each network segment must be sufficiently high for acceptable end-to-end bandwidth. This distinction is key to how we derive estimates from raw lower layer context to semantically match application QoS abstractions. Semantic context matching is elaborated in Sect. 6 (Table 1).

5.2 Lower-layer context

Basic physical layer data are known natively through the CR/DSA MAC-PHY parameters. They include the set of channels C (or the *spectrum*)[2] available for DSA, and the corresponding access protocols to use in each spectrum region. Consider any channel $c \in C$. The overall channel utilization factor is represented by $u(c)$, which is the average fraction of time that the channel c was used for communication. If T_{ON} represents the total duration for which the channel saw activity over a period of time T, then $u(c) = T_{ON}/T$. The average channel utilization by PUs ($u^{pu}(c)$), and by SUs ($u^{su}(c)$), is also known through the SOM via spectrum sensing (as discussed in Sect. 3). The total utilization seen on channel c is therefore:

$$u(c) = u^{pu}(c) + u^{su}(c) \quad (1)$$

Average ON/OFF durations of PUs ($E[T_{on}^{pu}(c)]$, $E[T_{off}^{pu}(c)]$) and SUs ($E[T_{on}^{su}(c)]$, $E[T_{off}^{su}(c)]$) on channel c are also known, as they are used to calculate $u^{pu}(c)$ and $u^{su}(c)$ [34].

The expected spectrum sensing duration ($t_{sense}(c)$) and the rate of sensing schedule ($r_{sense}(c)$) are obtained from spectrum scheduling scheme of DSA. The values of r_{sense} and t_{sense} are chosen from a range determined by the PHY sensing mechanisms and regulatory policies. Also, the set of their allowed values can be different for different channels in the spectrum.

In contrast to the above mentioned MAC-PHY parameters, some of the required lower layer information may not be directly available. In such cases, low overhead estimation techniques are used, as discussed next.

The physical layer data-rate ($b(c)$) and the application layer efficiency ($e(c)$) for each of the MAC-PHY scheme on channel c is assumed to be known, as mentioned in Sect. 2.3 $b(c)$ is the raw bit-rate that the MAC-PHY protocol can support, which depends on factors like modulation, encoding/decoding schemes, etc. If the MAC-PHY protocol has a dynamic rate adaptation feature, then recent historical information on data-rate employed will be used to compute the weighted average estimate for $b(c)$. Weights are based on the duration for which the data-rate was used. If no history is available yet, a median of available data-rates is selected as the initial value. The efficiency value $e(c)$ represents the average useful fraction of the raw bit-rate $b(c)$ available on channel c (from an application's viewpoint). It captures the overhead introduced into the communication stream by lower layer processing (e.g., link/physical layer protocol headers).

λ_{pkt} is the cumulative rate of packets arriving at link layer transmission queue from all the applications running on the device. As noted in Sect. 2.3, λ_{pkt} can be safely assumed to be constant across the short epoch (τ_{casa}) duration.

$t_{switch}(c1, c2)$ denotes the average time for the wireless interface to enter the ready state for data transfer in a new channel $c2$ after switching from the current channel $c1$. It includes the time to deactivate/activate new MAC-PHY mechanisms (if needed), as well as, associating with the new secondary service gateway or base station. If $t_{switch}(c1, c2)$ values are not available statically beforehand (for any $c1$, $c2$ pair), CASA builds this information progressively

[2] "Channel" implies "licensed channel", unless otherwise mentioned.

from recent historical observations by calculating their average.

6 Semantic alignment of contexts

In order to adapt DSA to application needs there must be a semantic congruence between application QoS abstractions and lower-layer DSA attributes that impact them. Albeit limited to application and MAC/PHY layers in this paper, we believe that semantic context alignment is a significant step towards realizing fully cognitive networks [39, 40]. Note, however, that the concept of semantic matching introduced here is at a high level and generic. For example, it is different from predicting the expected share of a channel's airtime, as in [24].

To achieve the semantic matching of application and lower-layer context, we process the raw MAC-PHY information to provide a reasonable estimate of attainable application-layer QoS. As noted in our design principles (Sect. 4), we trade some accuracy for low overhead and implementable system design. Thus, we avoid restrictive assumptions and complicated modeling. Instead, we employ average values and approximate estimates in translating low level context to compare with application-level QoS abstractions.

$B(c)$ denotes the average application-layer bandwidth expected on channel c, which is estimated as follows.

$$B(c) = b(c).e(c).\{1 - u(c)\}. \tag{2}$$

$B(c)$ is the effective throughput of useful data (i.e., application traffic) expected on channel c, when DSA is active.

Delay and jitter are *transit-additive* parameters, as mentioned in Sect. 5. Therefore, we calculate the "additional" delay and jitter contributed by the DSA network, rather than their overall end-to-end values. In our system model, only the one-hop wireless link behavior changes due to DSA, while other aspects of the end-to-end traffic flow remain the same.

Since PUs repeat an ON/OFF cycle of channel access, it is sufficient to consider a unit cycle of average duration $T_{cycle}(c) = E[T^{pu}_{on}(c)] + E[T^{pu}_{off}(c)]$ for channel c.

To calculate the additional delay, we take into account the following observations. First, communication is stopped during PUs' ON durations. Second, communication is delayed when the channel is shared with other SUs. In the worst case, a SU can be delayed for the entire duration during which other SUs access the channel. Third, communication is stopped during sensing periods. Finally, during the rest of the time, $T_{free} = T_{cycle} - (E[T^{pu}_{on}(c)] + E[T^{su}_{on}(c)])$, the communication continues without additional delay due to DSA. Therefore, the average additional delay is given as:

$$\begin{aligned} D(c) =& \frac{1}{\lambda_{pkt} T_{cycle}} \Big\{ (\lambda_{pkt} E[T^{pu}_{on}(c)]).\frac{E[T^{pu}_{on}(c)]}{2} \\ &+ (\lambda_{pkt} E[T^{su}_{on}(c)]).\frac{E[T^{su}_{on}(c)]}{2} \\ &+ \lambda_{pkt}.(t_{sense}(c) r_{sense}(c) T_{free}).\frac{t_{sense}}{2} + 0 \Big\} \\ =& u^{pu}(c).\frac{E[T^{pu}_{on}(c)]}{2} + u^{su}(c).\frac{E[T^{su}_{on}(c)]}{2} \\ &+ \{1 - u(c)\} r_{sense}(c).\frac{t^2_{sense}}{2}. \end{aligned} \tag{3}$$

In the above equation, $E[T^{pu}_{on}(c)]/2$ and $E[T^{su}_{on}(c)]/2$ are approximate estimates of the average extra delay when incumbents and other secondary devices access the channel. Sensing impact on delay, when the traffic is already stopped due to incumbent access or sharing, is not considered. Note that $\lambda_{pkt} T_{cycle}$ is the average number of application packets generated during the entire cycle for channel c.

To derive the additional jitter, again consider a cycle of duration $T_{cycle}(c)$. Additional jitter is typically introduced due to the difference in delays encountered when the channel is occupied by incumbents and other SUs, as compared to a completely free channel. We use the standard mean packet-to-packet delay variation (MPPDV) metric, which is the basis of jitter calculation in RTP/RTCP as defined in RFC 3550 [37]. Note that the MPPDV jitter formula is defined from an end-to-end perspective, and is not additive in strict mathematical sense. However, here we are concerned with "transit-additive" nature of jitter along intermediate hops. Thus, the average extra jitter is given by:

$$\begin{aligned} J(c) =& \frac{1}{\lambda_{pkt} T_{cycle}} \Big\{ (\lambda_{pkt} E[T^{pu}_{on}(c)]).\frac{1}{\lambda_{pkt}} \\ &+ (\lambda_{pkt} E[T^{su}_{on}(c)]).\frac{1}{\lambda_{pkt}} \\ &+ \lambda_{pkt}.(t_{sense}(c) r_{sense}(c) T_{free}).\frac{1}{\lambda_{pkt}} + 0 \Big\} \\ =& \frac{1}{\lambda_{pkt}} \{u(c) + (1 - u(c)) t_{sense}(c) r_{sense}(c)\}. \end{aligned} \tag{4}$$

We observe that bandwidth and additional delay estimates (Eqs. (2) and (3)) do not depend on packet-arrival rate λ_{pkt}. All the derivations above are based on uniform application packet arrival (which is valid considering the short epoch duration and QoS-sensitive traffic characteristics), but it can be shown that the semantic dependency equations (2), (3), and (4) are the same for other distributions like the Poisson packet-arrival process.

As seen from our experiment results in Sect. 10.3.3, the semantic context matching equations (2), (3), and (4) are sufficiently accurate in estimating the higher-layer QoS abstractions from lower-layer information.

7 CASA Algorithm

CASA Algorithm (see Algorithm 1) executes at the beginning of every CASA Epoch (τ_{casa}), after $B(c)$, $D(c)$ and $J(c)$ values are updated using the semantic dependency equations. At an abstract level, CASA Algorithm is a greedy reward-based algorithm and is loosely derived from Reinforcement Learning (RL) techniques, e.g., Q-Learning [14], in which rewards are assigned to actions based on past history of success.

RL techniques have been previously found to be directly useful in DSA optimization, e.g., for channel selection and spectrum sharing [13, 41]. Though we build upon the well-studied RL approach, CASA Algorithm is not a direct adaptation of any RL algorithm. Further, unlike such prior works in DSA, CASA Algorithm is utilized for optimizing DSA operations in relation to application QoS parameters.

CASA Algorithm strives to move DSA to a state where all application requirements are met by giving higher rewards to channels and DSA parameter combinations that achieve this goal to a higher degree. The abstraction of rewards is captured through the *push factor* value, which is explained below.

The algorithm first initializes the current push factor P_{curr} to 0. The cumulative weight of previous push factors up to the last epoch (a history window of size N), is represented by P_{past}. We use simple exponential function to compute P_{past}. If 0 is the most recent epoch and $N-1$ the least recent, then $P_{past} = \sum_{i=0}^{N-1} P_i w^i$, where w is the positive non-zero weight unit ($w < 1$). Clearly, the push factor in the most recent epoch has the highest weight and decreases for epochs in less recent epochs. This computation scheme captures the short-term correlation of near future with recent past in terms of the networking state experienced by the SU device. Note that $P_{past} + P_{curr}$ should be bounded above by P_{MAX}, and hence, P_{MAX} must be chosen based on window size N and weight unit w.

After the initialization step, CASA Algorithm performs multiple checks for potential application requirement violations and takes decisions to adapt DSA operations. In the first two decision-making steps, the algorithm adjusts r_{sense} and t_{sense} on the current channel, if feasible, in order to meet the delay and jitter requirements (D_{req}^{app} and J_{req}^{app}). This prevents violation of these requirements during spectrum sensing.

In addition to adjusting DSA parameters, the push factor P_{curr} is concurrently updated by checking the extent to which the application requirements are satisfied. Any shortfall in meeting a requirement increases the push factor. Based on the combined push factor, $P_{curr} + P_{past}$, if there is a better channel available, CASA goes for a channel-switch in a probabilistic fashion. A random value in [0,1] is generated, and a channel-switch occurs if the generated value is less than $(P_{curr} + P_{past})/(P_{MAX} + 1)$.

Algorithm 1 CASA Algorithm

Require: $(B_{req}^{app}, B_{obs}^{app})$, $(D_{req}^{app}, D_{obs}^{app})$, $(J_{req}^{app}, J_{obs}^{app})$
Require: $C, \forall c \in CB(c), D(c), J(c), t_{sense}(c), r_{sense}(c)$
Ensure: C is sorted in increasing order of $u(c)$
1: Let $chan \leftarrow$ current channel
2: Calculate P_{past}
3: $P_{curr} \leftarrow 0$ {Initialize current push factor}
4: **if** $B_{obs}^{app} < B_{req}^{app}$ **then**
5: $P_{curr} \leftarrow P_{curr} + 1$
6: **end if**
7: **if** $D_{obs}^{app} > D_{req}^{app}$ **then**
8: $P_{curr} \leftarrow P_{curr} + 1$
9: Reduce $r_{sense}(chan)$, $t_{sense}(chan)$, if possible
10: **end if**
11: **if** $J_{obs}^{app} > J_{req}^{app}$ **then**
12: $P_{curr} \leftarrow P_{curr} + 1$
13: Reduce $r_{sense}(chan)$, $t_{sense}(chan)$, if possible
14: **end if then**
15: **if** $(P_{curr} + P_{past}) > 0$ **then**
16: **for** each $c \in C, c \neq chan$ **do**
17: **if** $B(c) \geq (1/\gamma_B) B_{req}^{app}$ **then**
18: $P_{curr} \leftarrow P_{curr} + 1$
19: **end if**
20: **if** $D(c) \leq \gamma_D D_{req}^{app}$ **then**
21: $P_{curr} \leftarrow P_{curr} + 1$
22: **end if**
23: **if** $J(c) \leq \gamma_J J_{req}^{app}$ **then**
24: $P_{curr} \leftarrow P_{curr} + 1$
25: **end if**
26: **if** $t_{switch}(chan, c) \leq \gamma_D D_{req}^{app}$
27: $P_{curr} \leftarrow P_{curr} + 1$ **then**
28: **end if**
29: **if** $(P_{curr} + P_{past})/(P_{MAX} + 1) > rand[0,1]$ **then**
30: Switch to channel c
31: break
32: **end if**
33: **end for**
34: **end if**

Clearly, a higher push factor implies better chances for a channel-switch, and vice versa. Probabilistic channel-switching prevents overcrowding a single channel, which could happen if multiple SU devices switch to a channel that is globally perceived to be the best at the moment. Note that this channel-switching invocation is independent of the deterministic channel-switching that may occur due to other DSA events. On every channel-switch, the device initializes the sensing parameters (r_{sense}, t_{sense}) to its highest possible values, so that sensing is most aggressive.

To allow for flexible operational control in practice, CASA Algorithm incorporates configuration parameters

for its comparison process. For instance, γ_D acts as an administrative control knob that determines allowable limits on additional delay relative to end-to-end delay bound, depending on the deployment environment characteristics. For instance, in realistic deployments, γ_D would typically be less than 0.2. Similarly, γ_J and γ_B are the control knobs for jitter and bandwidth comparisons, respectively.

In summary, N, w, P_{MAX}, γ_D, γ_J, and γ_B constitute the configurable design parameters. CASA Algorithm has very low runtime and space complexity—linearly proportional to size of set C, thus reflecting the design principle of providing low overhead and easily deployed application QoS adaptation for DSA. While we argue that CASA has a strong engineering design for effective tradeoff between accuracy and overhead when providing QoS management in DSA, it cannot achieve the near-optimal state under circumstances which do not have proper configuration parameter settings. Some initial training may be necessary to come up with best values for configuration parameters. For instance, the history window size (N) should be set according to the spectral characteristics of the spectrum region, and should also be based on the administrative necessity to make CASA more/less reactive. For example, if the spectral environment exhibits rapidly changing behavior (small PU ON/OFF time-scales), smaller N should be preferred to quickly purge out the history that may be inaccurate at the current moment.

In some scenarios, there could be additional traffic during an epoch, e.g., due to legacy applications that may not provide any information to CASA. We approximate their impact by again relying on the principle of correlation of future with recent past. The key idea is to monitor the MAC packet queue to calculate the exponential moving average of the actual bandwidth at link layer, and compare it with the expected B_{req}^{app} value. If actual bandwidth is found to be greater, then their ratio is added to γ_B, γ_D, and γ_J.

8 Cross-Layer Interaction Framework (CLIF)

For CASA to be effective, an efficient cross-layer mechanism is needed to access the application-layer context. This task is accomplished through CLIF (see Fig. 6), which exposes the interface for applications to export their hints. CLIF also provides low overhead maintenance of the current state of application context. Such state maintenance of context is necessary because CASA takes an epoch-based approach to adaptation, rather than instantaneous adaptation (which will have very high operational overhead), due to continuous variability expected at lower layers during DSA.

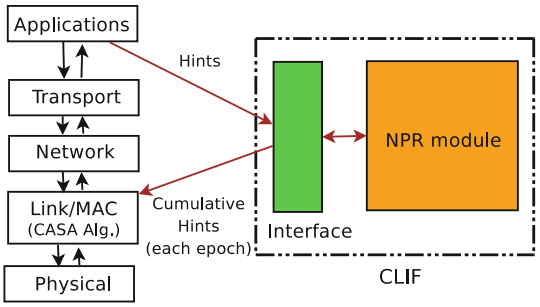


Fig. 6 Schematic overview of CLIF in relation to the network stack

CLIF consists of two components: the *Network Parameters Repository* (NPR), and the *Interface Functions*.

Network Parameters Repository is a central storehouse of parameters exported by the application layer. The link layer can query this module to get the parameter values, and hence, gain additional context information. NPR stores the parameters in a two dimensional hash table for fast lookups and updates. This module also incorporates a processing component in order to calculate basic aggregation functions, e.g., adding up individual bandwidth requirements to determine B_{req}^{app}.

Interface Functions consists of a well-defined list of functions (Fig. 7) for accessing and updating the NPR.

9 Implementation

Figure 8 shows our implementation model of CASA. We build CASA on an implementation of the abstract function model of DSA (described in Sect. 3). CLIF is implemented as a loadable Linux kernel module. Applications link with a user-level library implementation of the CLIF's Interface Functions. The CLIF implementation also provides an *Application Adaptation Layer* (AAL) to simplify interaction with multiple applications. CLIF has been implemented in the kernel-space rather than user-space, in order to improve performance by reducing user-kernel boundary crossings. This is because the frequency of access to CLIF module by lower-layer-resident CASA Algorithm is expected to be significantly higher than the rate of application context exports/updates.

DSA-based wireless interfaces are not yet commercially available as DSA protocols are still undergoing

```
ExportParameter(IN layer_id,IN param_id,IN param_value)
ImportParameter(IN layer_id,IN param_id,OUT param_value)
ImportParameter(IN param_id,IN op_type,OUT param_value)
UpdateParameter(IN layer_id,IN param_id,IN param_value)
RemoveParameter(IN layer_id,IN layer_id,IN param_id)
GetParameterList(OUT param_ids)
```

Fig. 7 Important *Interface Functions*

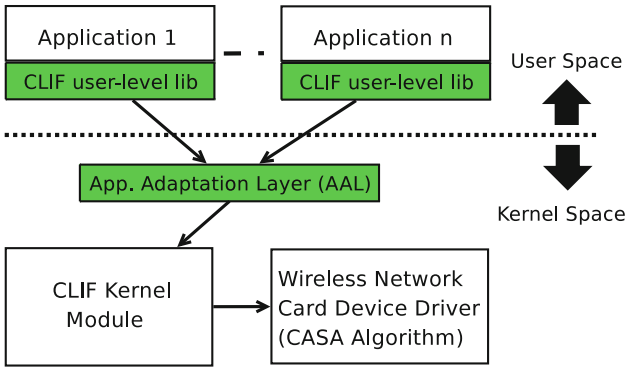


Fig. 8 Implementation model of CASA

standardization process. Further, we lack the license to transmit in the licensed spectrum. Hence, we use 802.11 protocol as the base MAC protocol to implement and test the CASA framework. We use Atheros cards (specifically, Linksys WPC55AG cards) in 5.2 GHz band (802.11a) with the MadWifi driver [38] in Ubuntu Linux based on 2.6.24-23 preemptible kernel.

Modifications were made to the MadWifi driver code to emulate periodic sensing (i.e., scheduling quiet periods). Incumbent detection is accomplished by reserving specific ESSIDs only for the incumbent networks. A channel-switch involves changing to a new 802.11a channel and associating with the the network on the new channel. For incumbent nodes, several changes are introduced in the MadWifi code to emulate the PUs' behavior, e.g., carrier sense was turned off by disabling Transmit Opportunity (or TXOP), as well as, by utilizing transmission power asymmetry between PU and SU networks. *Click* modular router [42] was used in conjunction with MadWifi in incumbent nodes to generate short duration ON/OFF periods with sufficient accuracy. We also had to resort to high resolution timers (`hrtimers`) in the kernel to provide interrupts with millisecond precision.

10 Evaluation

10.1 Evaluation metrics

We use the term "state-of-art DSA" to denote a DSA protocol that does not utilize the CASA service architecture. We compared the performance of DSA operating with CASA against state-of-art DSA in terms of the following metrics.

- Average application throughput (i.e., goodput), delay, and jitter during an application session run.
- Bandwidth fulfillment quotient (F_b)—fraction of session time during which the application bandwidth requirement is met.
- Delay fulfillment quotient (F_d)—fraction of session time during which the application delay requirement is met.
- Jitter fulfillment quotient (F_j)—fraction of session time during which the application jitter requirement is met.

Since the ultimate goal of DSA is to improve application performance, our metrics are application-centric, not of a lower-layer focus.

There are two important overheads introduced by CASA.

- Cross-layer communication delay (*CD*).
- Possible suboptimal channel selection and switches (*CS*).

CD occurs due to additional cross-layer communication related to export of application hints.

CS is introduced because CASA adjusts r_{sense} and t_{sense} as well as control channel-switch decisions in response to application requirements. A reduction in these parameters lead to stale information about channels, and may lead to poorer spectrum management decisions in DSA.

10.2 Testbed setup

Our experimental setup consists of AP-client laptop pairs—each pair is a part of 802.11 infrastructure-type first/last-mile access network. In each experiment, at least two secondary networks and one primary network are active. Every secondary network is composed of 3–5 clients paired with an AP. The test AP (i.e., the AP of the secondary network being tested) is connected to the Internet via our University ethernet. At application level, the test client connects to a fixed host in the wired segment through the test AP. This setup is in accordance with our system model (see Sect. 2).

We deploy the fixed host on a different subnet of the university LAN in an attempt to increase the number of hops in the end-to-end communication path. However, the machines are part of the same campus network—resulting in lower end-to-end latency and jitter compared to those typical on the open Internet. Nevertheless, this setup is adequate for testing CASA—the observed trends and insights remain valid for the generic DSA deployments. Figure 9 shows the testbed topology.

We mainly use `iperf` to generate traffic on secondary devices and to record the performance metrics. A custom script issues `iperf` commands and uses the CLIF user-level library to issue application hints. The traffic requirement mirrors the typical requirement of QoS-sensitive networking applications (e.g., VoIP/video streaming) [23, 43]. In particular, the bandwidth requirement is kept at 10 Mbps which reflects the traffic demands of many simultaneously running high-quality multimedia communication applications. Also,

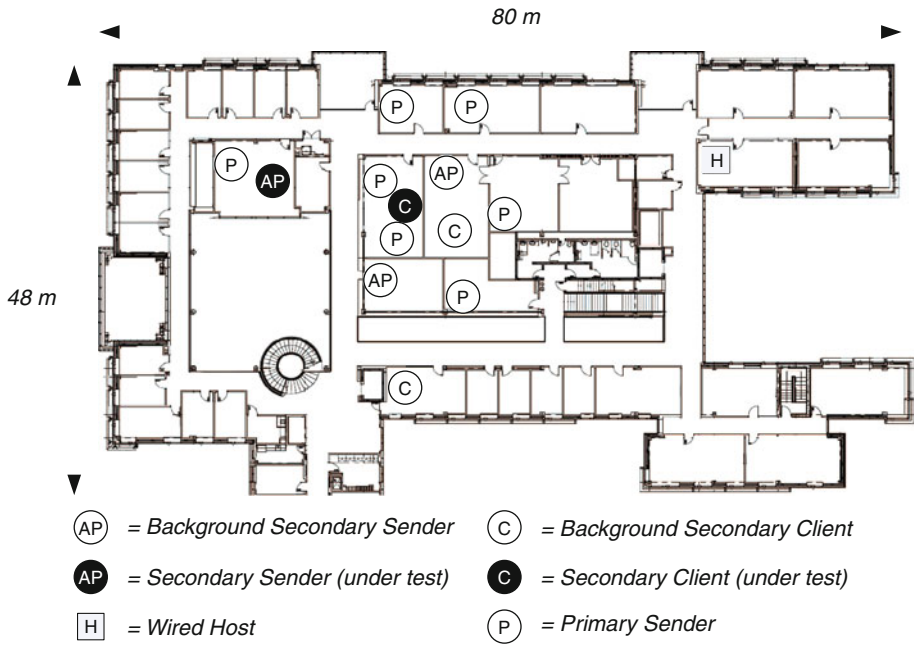


Fig. 9 The testbed is deployed on 4th floor of the department building. Each primary network is on a different channel. Only important nodes are shown to reduce clutter in the figure

we use delay requirement as 50 ms, and jitter requirement as 2 ms, based on typical single network SLA agreements [44]. Further, to compare actual consumer application performance with/without CASA, we use the open-source Ekiga softphone [45] (formerly known as GnomeMeeting) to generate video-conferencing sessions.

The spectrum C consists of seven channels (802.11a channels 36, 38, 40, 42, 44, 46, and 48). There are seven incumbent networks, one on each channel. They generate different random ON/OFF traffic with the average $T_{ON} + T_{OFF} = 50$ ms. Additional secondary devices are also enabled on the channels such that the average spectrum availability varies across the spectrum—70, 60, 50, 40, 30, 20, and 10% utilization on channels 36, 38, 40, 42, 44, 46, and 48, respectively. The raw physical-layer capacity is 54 Mbps for channels 36–44, while it is reduced to two-third and one-third of 54Mbps for channels 46 and 48, respectively.

This spectrum setup emulates the expected environment for DSA networking, where certain licensed channels exhibit higher utilization (e.g., because of lower DSA access-cost/physical capacity ratio) while others have lower utilization (e.g., because of higher DSA access-cost/physical capacity ratio). Note that secondary market business model will play a major role in governing DSA networks, but they are still under active development [9, 46, 47] and beyond the scope of this paper.

For emulating state-of-art DSA, we use $r_{sense} = 2$ (per second), and $t_{sense} = 0.05$ s. We use the following CASA parameter values for all channels (unless otherwise noted):

(1) $r_{sense} \in \{4, 2, 1, 0.5, 0\}$ (per second).
(2) $t_{sense} \in \{0.1, 0.05, 0.025, 0.0125, 0\}$ (second).
(3) $\tau_{casa} = 1$ (second).
(4) $w = 0.5$, and $P_{MAX} = 6$.
(5) $N = 8$ epochs.
(6) $\gamma_J = 0.2$, $\gamma_D = 0.2$, $\gamma_B = 0.5$

DSA parameter values described here are based on their typical values in standard drafts, e.g., 802.22 [22]. Additional discussion on selecting the values for r_{sense} and t_{sense} can be found in [29, 34]. Values for CASA-related configuration parameters (e.g., τ_{casa}) have been chosen keeping in view the values of key DSA parameters and through calibration from initial experiments.

10.3 Results and discussion

10.3.1 Overhead analysis

We characterize the overhead *CD* through a timing study of CLIF system calls in the Linux kernel. The average delay with CLIF interface function calls is observed to be around 1 µs on average. This is insignificant compared to acceptable network delays and traffic-burst time (which are in the order of tens of ms) of applications, especially as there are only a few interface function calls associated with each application communication session. Further, the additional memory space taken by CASA is found to be quite low on average (around 20 kB, with 3 kB SD) throughout our evaluation. Note that these averages are calculated from observations across all the experiments outlined in Sects. 10.3.3–10.3.5. Thus, they include the impact of variation across several test factors, like spectrum size, incumbent ON/OFF duration, application type, etc. As seen later, we can conclude that CASA's overhead

is negiligible in comparison to the performance gains acheived.

10.3.2 Accuracy of DSA implementation

To test that DSA features are properly implemented (e.g., primary traffic must not be interfered) on base 802.11 MAC, we conducted simple microbenchmark experiments. In this experimental setting, we have one primary and one secondary network and the spectrum is limited to one channel (c = channel 36), with PHY capacity set at 24 Mbps. Both the primary and secondary transmitters are saturated with UDP traffic, i.e., they always have packets to transmit. We allow the primary transmitter to operate at a specified channel utilization. The rest of the channel-time is available for opportunistic secondary utilization. Each experiment run lasts for 2 min, and we conducted 100 test runs. Figure 10 shows the average throughput plotted against primary traffic load. As seen from the graph, the primary traffic gets preferential channel access, as it should, when DSA is deployed. The primary throughput is limited only by its own channel utilization. When more opportunities are available on the channel, the secondary network (both for state-of-art DSA as well as CASA-based DSA) shows greater throughput, as expected. Thus, apart from showing that our implementation is accurate, the result also shows the impact of primary traffic load on secondary communication.

Even in this simple scenario, using CASA results in better performance than state-of-art DSA, especially at higher primary utilizations. The throughput improvement is around 8% when PU utilizatin is 0.4, while the improvement margin increases to 51% when PU utilization is 0.8. This outcome is due to CASA dynamically managing DSA parameters to accommodate application traffic needs without violating DSA's constraints. Note that this improvement can be even better if the spectrum consists of more channels—this is shown in results discussed later.

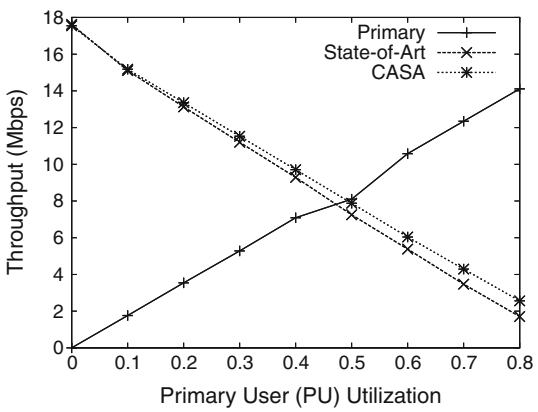


Fig. 10 Throughput with variation in primary traffic volume

Thus, CASA is found to improve application performance for DSA and also make it more resilient.

10.3.3 Accuracy of semantic matching

Semantic dependency equations are microbenchmarked in a controlled environment to ascertain their validity in terms of their estimation accuracy. For this purpose, we again use a single channel (c = channel 36) with one secondary network as the test candidate. The PHY-layer capacity is set at 54 Mbps. Both primary and background secondary transmitters are present on the channel. The total utilization $u(c)$ is changed for each set of experiments, with $u^{pu}(c)$: $u^{su}(c) = 4{:}1$ in each case.

Also, due to the *transit-reductive* nature of bandwidth parameter (see Sect. 5), saturation-level UDP traffic is used to quantify the maximum bandwidth available for different values of $u(c)$. For the delay/jitter case, we use UDP traffic at 10 Mbps, with packet size of 500 bytes ($\lambda_{pkt} = 2500$). Further, $e(c) \approx 0.6$ for 802.11 MAC using UDP. CASA Algorithm is disabled—only semantic matching is performed. The sensing parameters, therefore, are fixed at $r_{sense} = 2\ s^{-1}$ and $t_{sense} = 0.05$ s.

Figure 11 shows that the observed values from experiments closely match the predictions from the semantic dependency equations. For each of bandwidth, delay, and jitter parameters, the average observed values (with 95% confidence interval) are within 8% of their predictions. Thus, the proposed semantic matching is found to be highly accurate. For higher channel utilizations ($\approx >0.7$), the predicted values diverge slightly from the observed values. This arises primarily due to unaccounted MAC effects that are significantly more prominent at a very high contention level, e.g., unusually large backoff duration. However, the observed difference is very small, and further, very high utilization channels are unlikely to be used for DSA.

10.3.4 CASA performance

To analyze the end-to-end performance of CASA, we use the setup described earlier in Sect. 10.2. Figure 12 shows the temporal variation for throughput, delay and jitter in a 30 s period starting from the beginning of the session. Each point on the graphs denotes the average value over the past 1 s window. Note that we measure the QoS metrics for several minutes, but skip the full duration in the plots as the long-term behavior is similar.

As seen from the graphs, CASA-based DSA produces significant improvements for each metric. In particular, throughput is found to dip starting at around 8 s mark for state-of-art DSA. This corresponds to a channel-switch event to channel 48. State-of-art DSA is found to switch to a channel with lower utilization without considering

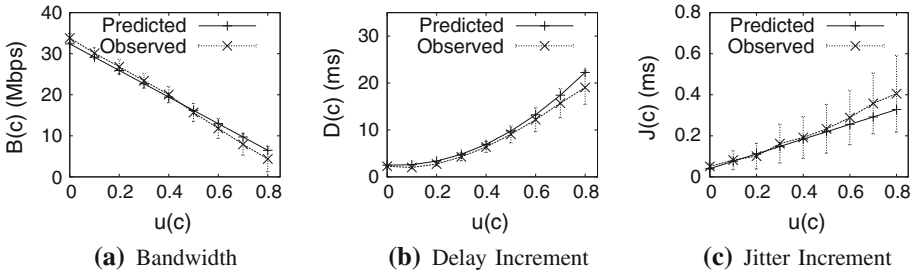


Fig. 11 Analyzing the accuracy of semantic context matching equations

(a) Bandwidth **(b)** Delay Increment **(c)** Jitter Increment

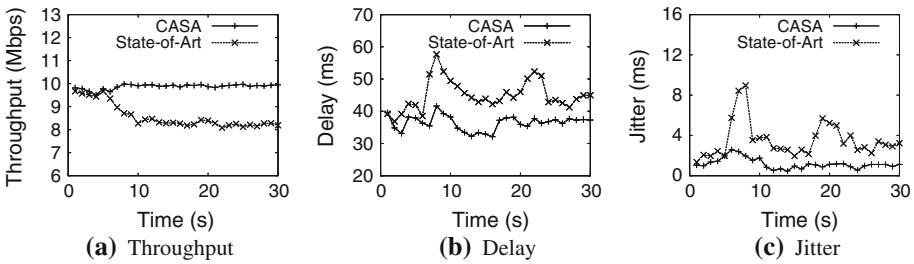


Fig. 12 Temporal variation of QoS metrics

(a) Throughput **(b)** Delay **(c)** Jitter

the expected QoS performance in the new channel. Thus, the throughput drops as the new channel cannot support the 10 Mbps bandwidth. With CASA, while the secondary group still switches channels at approximately the same time, the new channel is selected such that it can support the application's bandwidth demand. Consequently, the throughput requirement is continued to be met throughout.

The end-to-end delay and jitter (Fig. 12b, c) is also found to be higher (with considerably more variation) for state-of-art DSA. Again, the reason for this is that CASA Algorithm results in a better channel selection, where packets are delayed less and the variability is lower. Adaptation of sensing parameters also contributes towards achieving better delay/jitter performance.

Figure 13 shows the normalized values (together with the 95% confidence interval) for the QoS metrics obtained relative to their requirements, for both state-of-art and CASA-based DSA. The observations are made over a 120 s duration. Also, Fig. 14 shows the fulfillment quotient of the QoS demands (F_b, F_d, and F_j) over the same duration.

From the two graphs, CASA-based DSA outperforms state-of-art DSA in fulfilling the requirements, especially in supporting QoS demands through the full duration of the communication session. For example, the throughput for CASA is $\approx$10Mbps (Fig. 13), as required, for more than 90% of the communication session (Fig. 14). Similarly, average delay and jitter are 30 and 64%, respectively, lower than the requirement for more than 90% of the session run time. CASA is observed to be especially effective in reducing jitter to a very low level as compared to state-of-art DSA. Thus, even the stringent jitter requirement of 2 ms is satisfied for almost the entire session.

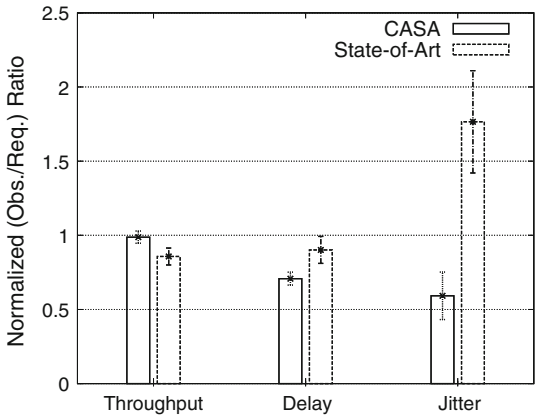


Fig. 13 Performance comparison in terms of average values of QoS parameters. Observed values normalized w.r.t. required values are plotted

State-of-art DSA is also found to be somewhat effective in matching up to the delay metric. We attribute this to the topology bias of our testbed. In particular, the network path traversed by the packets is quite small (the end-hosts lie on the same network domain) with very few intermediate hops. Thus, the end-to-end delay typically does not exceed the application-mandated requirement despite DSA side-effects. In general, we expect state-of-art DSA to perform significantly worse in the open Internet scenario, where the end-to-end delay is significantly higher. Thus, the benefits of CASA would be more pronounced than that demonstrated in our setup.

To study the performance of consumer-oriented applications, we run 6 videoconferencing sessions on the test secondary network using Ekiga VoIP softphone [45]. Each videoconferencing session requires around 460kbps (G.711

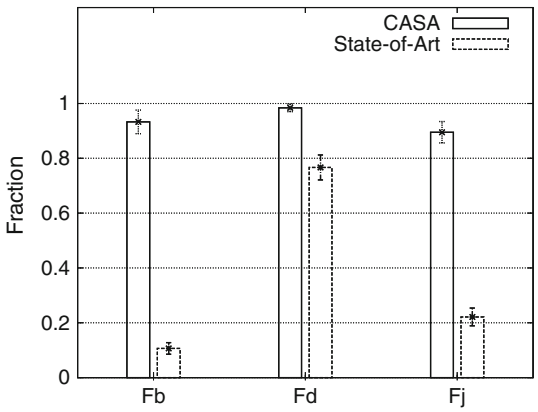


Fig. 14 CASA is effective in sustaining QoS over the entire communication session

audio with H.261 video codecs used) of application throughput. Figure 15 shows the comparison of performance (on QoS metrics) when CASA is used as compared to state-of-art DSA. With CASA enabled, the VoIP session is able to consistently achieve the required goodput demand—an improvement of 33% over state-of-art DSA. Similarly, the delay and jitter found to reduce by 28 and 67%, respectively—thus illustrating CASA's ability to match up to QoS demands in realistic settings.

10.3.5 Effects of design parameters

Experiments were also conducted to study the impact of CASA's design parameters. In brief, following are the key observations.

Increasing P_{MAX} reduces channel-switches, and hence, applications do not get sufficient bandwidth in situations where a channel is shared by a large number of secondary users.

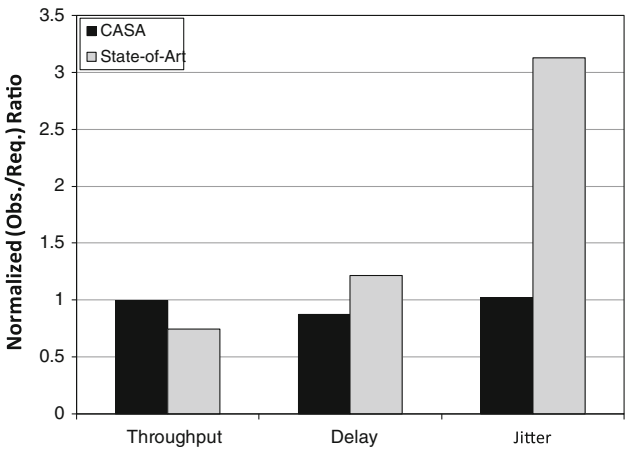


Fig. 15 Performance comparison with Ekiga VoIP sessions. Observation is normalized w.r.t. to requirements

Increasing the history window size N (beyond the current value of 8) does not lead to any significant change in the observed behavior. However, reducing N results in frequent channel-switches (especially when $N \leq 2$) leading to poorer performance on all metrics.

The effect of changing the epoch duration (τ_{casa}) is found to be similar to that of modifying N.

10.3.6 Summary of evaluation results

As is evident from the evaluation results, the overheads *CD* and *CS* (which are implicitly incurred in all the experiments) do not offset the advantages of CASA. The application-centric context-awareness introduced by CASA improves application performance significantly. Average delay and jitter are improved by 30% on average. Also, CASA fulfills QoS demands for more than 90% of the duration of a communication session, which is more than triple of state-of-art DSA. Hence, CASA increases the resilience of DSA protocols in supporting application demands to a greater degree, especially in unfavorable environment.

11 Related work

There have been several proposals for DSA protocols in literature [22–26, 48, 49], as well as, their component technologies, e.g., spectrum sensing [33, 34, 50], spectral resource allocation [51], and cognitive radios [52, 53]. Though all of these prior work incorporate channel-awareness, they do not consider DSA's side-effects on specific application-layer QoS requirements.

Certain recent works attempt to account for QoS and context-awareness in DSA [10, 12, 13] using either Reinforcement Learning or Game Theory. However, the QoS considered in these proposals is not the high level application QoS metrics, but link level metrics like SNR and BER [10] or number of successful link layer transmissions [13]. Clearly, the main drawback of these papers is their narrow QoS context which results in coarse-grained "one-size fits all" solution. This approach does not work in realistic situations given DSA's side-effects identified in our work. Network applications and services (like VoIP) have QoS requirements which are not defined in terms of such low level parameters. Application QoS demands (like goodput, delay, and jitter) have higher level semantics and their dependency on link level parameters is not considered in such prior works, which we rigorously define through proposed semantic matching in CASA. Consequently, CASA is able to provide effective, fine-grained, and dynamic adaptation to application QoS demands during DSA operation.

Apart from narrow QoS context, the adaptive responses in [10, 12, 13, 54] are also limited and just include channel selection and spectrum sharing schemes—typically for every packet transmission which incurs very high overhead. No adaptation of DSA's fundamental parameters (like sensing duration, etc) is considered. Finally, the implementation and practical issues are not analyzed or evaluated. On the other hand, our prototype implementation and evaluation shows that CASA can be easily deployed, and has low operational overhead while providing significant QoS benefits to consumer applications in realistic settings.

Our approach is inspired by the benefits shown for adaptation based on application behavior/requirements in several other system optimization strategies, including power management of wireless interfaces [15] and wireless network selection [16]. In principle, CASA adopts a similar approach, providing a cross-layer framework in order to provide necessary application context to the underlying DSA protocol which can then adapt intelligently (through CASA Algorithm) to accommodate application QoS requirements.

The necessity of upper-layer assimilation with DSA has been pointed out in [2] and the associated QoS issues have been studied in [36, 55]. There has been some work on higher-layer adaptation given the presence of heterogeneous wireless technologies at lower layer [56]. A new transport protocol, called TP-CRAHN, has been proposed in [31] to address TCP performance issues when DSA is deployed. Transport layer impact of DSA is also studied in [30] in context of end-to-end connections. Traffic load has been used as the key metric in designing adaptive spectrum management schemes in [57] and [58]. Both of these works target enterprise Wifi LANs (specifically, access points rather than client devices), and do not consider application QoS needs like jitter or delay. Further, they offer adaptive channel allocation/management as the only solution, rather than adaptation of DSA parameters.

12 Concluding remarks

12.1 Conclusion

We argued for, and showed the importance of application-awareness in making DSA intelligent and more robust. We proposed a systems-based optimization mechanism together with the service architecture, called *Context Aware Spectrum Agility* (CASA), that combines application-awareness with channel-state knowledge in DSA. CASA targets wireless client-type end-devices, and has been designed for easy deployment and implementation. The key component of CASA is the *CASA Algorithm*, which dynamically adapts DSA to accommodate specific application QoS requirements. We derived semantic dependency equations for matching application context with lower layer MAC-PHY information, which play a critical role in the CASA Algorithm. CASA can operate with any DSA protocol without introducing any significant modification or overhead. Our evaluation has shown CASA to increase the resilience of the DSA protocols significantly in supporting stringent application QoS requirements with intelligent, application-aware decisions. CASA illustrates that application-centric adaptivity of DSA parameters is effective in managing DSA side-effects, thereby enhancing the benefits of DSA in consumer wireless networks.

12.2 Future work

Our short-term goal is to observe CASA's behavior on different types of traffic distributions or traffic classes. Improving the "context" information given to CASA can lead to a further improvement in application performance when DSA is utilized. In particular, we intend to investigate the impact on packet loss rate due to DSA, and extend semantic matching to the loss metric so that it can be incorporated in the CASA service architecture. Further, we intend to provide support for priority classes in CASA. This will allow preferential treatments for satisfying QoS demands for different types of applications during DSA.

References

1. Kumar, A., & Shin, K. G. (2007). Extended abstract: Towards context-aware wireless spectrum agility. In *13th ACM MOBICOM* (pp. 318–321).
2. Akyildiz, I. F., Lee, W.-Y., Vuran, M. C., & Mohanty, S. (2006). Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *Computer Networks Journal, 50*(13), 2127–2159.
3. McHenry, M. A., & Steadman, K. (2004). *Spectrum occupancy measurements*. Shared Spectrum Company, http://www.sharedspectrum.com/measurement.
4. Chen, D., et al. (2009). Mining spectrum usage data: A large-scale spectrum measurement study. In *ACM MOBICOM* (pp. 13–24).
5. FCC. (2010). *Unlicensed operation in the TV broadcast bands: Second memorandum opinion and order*. ET Docket No. FCC 10-174.
6. Federal Communications Commission (FCC). (2008). *FCC adopts rules for unlicensed use of television white spaces*. FCC News 202/418-0500.
7. Shah, A. (2008). *Dell to offer 'white space' connectivity in laptops*. IT World, November 2008. http://www.itworld.com/mobile-amp-wireless/57281/dell-offer-white-space-connectivity-laptop.
8. Vort, J. (2009). *Microsoft, Dell, Spectrum Bridge launch first public white spaces*. Network World, October 2009. http://www.networkworld.com/community/node/4657.

9. Spectrum Bridge. *SpecEx: The Online Marketplace for Spectrum*. http://www.specex.com.
10. Zao, C., Jin, T., Chigan, C., & Tian Z. (2008). QoS-aware distributed spectrum sharing for heterogeneous wireless cognitive networks. *Computer Networks, 52*(4), 864–878.
11. Hamouda, S., & Hamdaoui, B. (2009). Dynamic spectrum access in heterogeneous networks: HSDPA and WiMAX. In *IEEE IWCMC* (pp. 1253–1257).
12. Pham, H. N., Xiang, J., Zhang, Y., & Skeie, T. (2008). QoS-aware channel selection in cognitive radio networks: a game-theoretic approach. In *IEEE GLOBECOM* (pp. 1–7).
13. Yau, K. L. A., Komisarczuk, P., & Teal P. D. (2010). Context-awareness and intelligence in distributed cognitive radio networks: A reinforcement learning approach. In *IEEE Communications Theory Workshop (AusCTW)* (pp. 35–42).
14. Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*, 1st edn. Cambridge, MA: MIT Press.
15. Anand, M., Nightingale, E. B., & Flinn, J. (2004). Ghosts in the machine: Interfaces for better power management. In *2nd ACM MOBISYS* (pp. 23–25).
16. Higgins, B., et al. (2010). Intentional networking: Opportunistic exploitation of mobile network diversity. In *16th ACM MOBICOM*.
17. *GNU Radio*. http://www.gnu.org/software/gnuradio.
18. Ettus Research LLC. *Universal Software Radio Peripheral*. http://www.ettus.co.
19. Haykin, S. (2005). Cognitive radio: Brain-empowered wireless communications. *IEEE JSAC* 23(2), 201–220.
20. Mitola, J., & Maguire, G. Q. (1999). Cognitive radio: Making software radios more personal. In *IEEE Personal Communications* (pp. 13–18).
21. USRP2. http://www.ettus.com/product.
22. IEEE 802 LAN/MAN Standards Committee 802.22 WG on WRANs. http://www.ieee802.org/22.
23. Szigeti, T., & Hattingh, C. (2004). *End-to-End QoS network design: Quality of service in LANs, WANs, and VPNs*, 1st edn. Indiana: Cisco Press.
24. Bahl, P., Chandra, R., Moscibroda, T., Murty, R., & Welsh, M. (2009). White space networking with Wi-Fi like connectivity. In *SIGCOMM* (pp. 27–38).
25. Hamdaoui, B., & Shin, K. G. (2008). OS-MAC: An efficient MAC protocol for spectrum-agile wireless networks. *IEEE Transactions on Mobile Computing, 7*(8), 915–930.
26. Cordeiro, C., & Challapali, K. (2007). C-MAC: A cognitive MAC protocol for multi-channel wireless network. In *2nd IEEE DySPAN* (pp. 147–157).
27. Zhao, Q., Tong, L., & Swami, A. (2005). A cross-layer approach to cognitive MAC for spectrum agility. In *IEEE Asilomar Conference on Signals, Systems and Computers*.
28. ECMA International (2009). *ECMA 392: MAC and PHY for Operation in TV White Space*. http://www.ecma-international.org/publications/standards/Ecma-392.ht.
29. Chou, C. -T., Kim, H., Shin, K. G., & Sai Shankar, N. (2007). What and how much to gain by spectral agility? *IEEE Journal on Selected Areas in Communications (JSAC), 25*(3), 576–588.
30. Kumar, A., & Shin, K. G. (2010). Managing TCP connections in dynamic spectrum access baded wireless LANs. In *7th IEEE SECON* (pp. 100–108).
31. Chowdhury, K. R., Di Felice, M., & Akyildiz I. F. (2009). TP-CRAHN: A transport protocol for cognitive radio ad-hoc networks. In *28th IEEE INFOCOM* (pp. 2482–2490).
32. Zhao, Q., Tong, L., Swami, A., & Chen Y. (2007). Decentralized cognitive MAC for opportunistic spectrum access in ad hoc networks: A POMDP framework. *IEEE Journal on Selected Areas in Communications (JSAC), 25*(3), 589–600.
33. Chang, N. B., & Liu, M. (2007). Optimal channel probing and transmission scheduling for opportunistic spectrum access. In *13th ACM MOBICOM* (pp. 27–38).
34. Kim, H., & Shin, K. G. (2008). Efficient discovery of spectrum opportunities with MAC-layer sensing in cognitive radio networks. *IEEE TMC, 7*(5), 533–545.
35. Kim, H., & Shin, K. G. (2008). Fast discovery of spectrum opportunities in cognitive radio networks. In *IEEE DySPAN* (pp. 1–12).
36. Kumar, A., Shin, K. G., Wang, J., & Challapali, K. (2009). A case study of QoS provisioning in TV-band cognitive radio networks. In *18th IEEE ICCCN* (pp. 1–6).
37. Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (2003). *RTP: A transport protocol for real-time applications. RFC 3550, internet engineering task force*. http://www.rfc-editor.org/rfc/rfc3550.tx.
38. *The MadWifi Project*. http://madwifi-project.org.
39. Mahmoud, Q. (2007). *Cognitive Networks: Towards self-aware networks*, 1st edn. New York: Wiley InterScience.
40. Thomas, R. W., DaSilva, L. A., & MacKenzie, A. B (2005). Cognitive networks. In *1st IEEE DySPAN* (pp. 352–360).
41. Yau, K. L. A., Komisarczuk, P., & Teal, P. D. (2009). A context-aware and intelligent dynamic channel selection scheme for cognitive radio networks. In *IEEE CROWNCOM*.
42. *Click Modular Router*. http://read.cs.ucla.edu/clic.
43. Xiao, X. (2008). *Technical, commercial and regulatory challenges of QoS: An internet service model perspective*, 1st edn. Los Altos, CA: Morgan Kaufmann.
44. Voip-Info.org. (2009). *VOIP QoS Requirements*. http://www.voip-info.org/wiki/view/Qo.
45. *Ekiga*. http://ekiga.or.
46. Nolan, K. E., et al. (2007). Value creation and migration in adaptive cognitive and radio systems. In *Cognitive radio, software defined radio, and adaptive wireless systems*, chapter 5 (pp. 145–159). Berlin: Springer.
47. Kim, H., & Shin, K. G. (2010). Understanding Wi-Fi 2.0: From the economical perspective of wireless service providers. *IEEE WCM, 17*(4), 41–46.
48. Deb, S., Srinivasan, V., & Maheshwari, R. (2009). Dynamic spectrum access in DTV whitespaces: Design rules, architecture and algorithms. In *15th ACM MOBICOM* (pp. 1–12).
49. Rahul, H., Edalat, F., Katabi, D., & Sodini, C. (2009). Frequency-aware rate adaptation and MAC protocols. In *15th ACM MOBICOM* (pp. 193–204).
50. Gandetto, M., & Regazzoni, C. (2007). Spectrum sensing: A distributed approach for cognitive terminals. *IEEE JSAC, 25*(3), 546–557.
51. Cao, L., & Zheng H. (2008). SPARTA: Stable and efficient spectrum access in next generation dynamic spectrum access networks. In *27th IEEE INFOCOM* (pp. 870–878).
52. Tan, K., et al. (2009). SORA: High performance software radio using general purpose multi-core processors. In *NSDI*.
53. Nychis, G., Hottelier, T., Yang, Z., Seshan, S., & Steenkiste, P. (2009). Enabling MAC protocol implementations on software-defined radios. In *NSDI*.
54. Chen, D., Zhang, Q., & Jia W. (2008). Aggregation aware spectrum assignment in cognitive ad-hoc networks. In *IEEE CROWNCOM*.
55. Ishibashi, B., Bouabdallah, N., & Boutaba, R. (2008). QoS performance analysis of cognitive radio-based virtual wireless networks. In *27th IEEE INFOCOM* (pp. 2423–2431)
56. Akan, O. B., & Akyildiz, I. F. (2004). ATL: An adaptive transport layer suite for next-generation wireless internet. *IEEE Journal on Selected Areas in Communications (JSAC), 22*(5), 802–817.

57. Moscibroda, T., Chandra, R., Wu, Y., Sengupta, S., Bahl, P., & Yuan, Y. (2008). Load-aware spectrum distribution in wireless LANs. In *IEEE ICNP* (pp. 137–146).
58. Yang, L., Cao, L., Zheng, H., Belding, E. (2008). Traffic-aware dynamic spectrum access. In *4th WICON*.

Author Biographies

Ashwini Kumar received his B.Tech. degree in Computer Science and Engineering from Indian Institute of Technology, Kanpur, India, in 2004. Currently he is a Ph.D candidate at University of Michigan. His research interests are QoS issues and resource management in wireless networks, including cognitive radio networks. He is a member of the IEEE and the IEEE Communications Society.

Kang G. Shin is the Kevin and Nancy O'Connor Professor of Computer Science and Founding Director of the Real-Time Computing Laboratory in the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, Michigan. His current research focuses on QoS-sensitive networking and computing as well as on embedded real-time OS, middleware and applications, all with emphasis on timeliness and dependability. He has supervised the completion of 65 Ph.D. theses, and authored/coauthored about 700 technical papers (more than 250 of which are in archival journals) and numerous book chapters in the areas of distributed real-time computing and control, computer networking, fault-tolerant computing, and intelligent manufacturing. He has co-authored (jointly with C. M. Krishna) a textbook "Real-Time Systems," McGraw Hill, 1997. He has received a number of best paper awards, including the IEEE Communications Society William R. Bennett Prize Paper Award in 2003, the Best Paper Award from the IWQoS'03 in 2003, and an Outstanding IEEE Transactions of Automatic Control Paper Award in 1987. He has also coauthored papers with his students who received the Best Student Paper Awards from the 1996 IEEE Real-Time Technology and Application Symposium, and the 2000 UNSENIX Technical Conference. He has also received several institutional awards, including the Research Excellence Award in 1989, Outstanding Achievement Award in 1999, Service Excellence Award in 2000, Distinguished Faculty Achievement Award in 2001, and Stephen Attwood Award in 2004 from The University of Michigan; a Distinguished Alumni Award of the College of Engineering, Seoul National University in 2002; 2003 IEEE RTC Technical Achievement Award; and 2006 Ho-Am Prize in Engineering. He received the B.S. degree in Electronics Engineering from Seoul National University, Seoul, Korea in 1970, and both the M.S. and Ph.D. degrees in Electrical Engineering from Cornell University, Ithaca, New York in 1976 and 1978, respectively. From 1978 to 1982 he was on the faculty of Rensselaer Polytechnic Institute, Troy, New York. He has held visiting positions at the U.S. Airforce Flight Dynamics Laboratory, AT&T Bell Laboratories, Computer Science Division within the Department of Electrical Engineering and Computer Science at UC Berkeley, and International Computer Science Institute, Berkeley, CA, IBM T. J. Watson Research Center, Software Engineering Institute at Carnegie Mellon University, and HP Research Laboratories. He also chaired the Computer Science and Engineering Division, EECS Department, The University of Michigan for 3 years beginning January 1991. He is Fellow of IEEE and ACM, and member of the Korean Academy of Engineering, is serving as the General Co-Chair for 2009 ACM Annual International Conference on Mobile Computing and Networking (MobiCom'09), was the General Chair for 2008 IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'08), the 3rd ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys'05) and 2000 IEEE Real-Time Technology and Applications Symposium (RTAS'00), the Program Chair of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chair of the 1987 RTSS, the Guest Editor of the 1987 August special issue of IEEE Transactions on Computers on Real-Time Systems, a Program Co-Chair for the 1992 International Conference on Parallel Processing, and served numerous technical program committees.He also chaired the IEEE Technical Committee on Real-Time Systems during 1991-93, was a Distinguished Visitor of the Computer Society of the IEEE, an Editor of IEEE Trans. on Parallel and Distributed Computing, and an Area Editor of International Journal of Time-Critical Computing Systems, Computer Networks, and ACM Transactions on Embedded Systems.