Mobile wireless applications are likely to benefit from cloud computing, as we discussed in Chapter 4.

This expectation is motivated by several reasons:

• The convenience of data access from any site connected to the Internet.

• The data transfer rates of wireless networks are increasing; the time to transfer data to and from a

cloud is no longer a limiting factor.

• Mobile devices have limited resources; whereas new generations of smartphones and tablet computers

are likely to use multicore processors and have a fair amount of memory, power consumption

is, and will continue to be, a major concern in the near future. Thus, it seems reasonable to delegate

compute- and data-intensive tasks to an external entity, e.g., a cloud.

The first application we discuss is a cloud-based simulation for trust evaluation in a Cognitive Radio

Network (CRN) [52]. The available communication spectrum is a precious commodity, and the objective

of a CRN is to use the communication bandwidth effectively while attempting to avoid interference

with licensed users. Two main functions necessary for the operation of a CRN are spectrum sensing

and spectrum management. The former detects unused spectrum and the latter decides the optimal

use of the available spectrum. Spectrum sensing in CRNs is based on information provided by the

nodes of the network. The nodes compete for the free channels, and some may supply deliberately

distorted information to gain advantage over the other nodes; thus, trust determination is critical for the

management of CRNs.

Cognitive Radio Networks. Research over the last decade reveals a significant temporal and spatial

underutilization of the allocated spectrum. Thus, there is a motivation to opportunistically harness the

vacancies of spectrum at a given time and place.

The original goal of cognitive radio, first proposed at Bell Labs [246,247], was to develop a softwarebased

radio platform that allows a reconfigurable wireless transceiver to automatically adapt its communication

parameters to network availability and to user demands. Today the focus of cognitive radio

is on spectrum sensing [58,161].

We recognize two types of devices connected to a CRN: primary and secondary. Primary

nodes/devices have exclusive rights to specific regions of the spectrum; secondary nodes/devices enjoy

dynamic spectrum access and are able to use a channel, provided that the primary, licensed to use that

channel, is not communicating. Once a primary starts its transmission, the secondary using the channel

is required to relinquish it and identify another free channel to continue its operation. This mode of

operation is called an overlay mode.

CRNs are often based on a cooperative spectrum-sensing strategy. In this mode of operation, each

node determines the occupancy of the spectrum based on its own measurements, combined with information

from its neighbors, and then shares its own spectrum occupancy assessment with its neighbors

[129,339,340].

Information sharing is necessary because a node alone cannot determine the true spectrum occupancy.

Indeed, a secondary node has a limited transmission and reception range; node mobility combined

with typical wireless channel impairments, such as multipath fading, shadowing, and noise, add to the

difficulty of gathering accurate information by a single node.

Individual nodes of a centralized or infrastructure-based CRN send the results of their measurements

regarding spectrum occupancy to a central entity, whether a base station, an access point, or a cluster

head. This entity uses a set of fusion rules to generate the spectrum occupancy report and then distributes

it to the nodes in its jurisdiction. The area covered by such networks is usually small since global spectrum

decisions are affected by the local geography.

There is another mode of operation based on the idea that a secondary node operates at a much lower

power level than a primary one. In this case the secondary can share the channel with the primary as long as its transmission power is below a threshold, μ, that has to be determined periodically. In this

scenario the receivers that want to listen to the primary are able to filter out the “noise” caused by the

transmission initiated by secondaries if the signal-to-noise ratio, (S/N), is large enough.

We are only concerned with the overlay mode whereby a secondary node maintains an occupancy

report, which gives a snapshot of the current status of the channels in the region of the spectrum it is

able to access. The occupancy report is a list of all the channels and their state, e.g., 0 if the channel is

free for use and 1 if the primary is active. Secondary nodes continually sense the channels available to

them to gather accurate information about available channels.

The secondary nodes of an ad hoc CRN compete for free channels, and the information one node may

provide to its neighbors could be deliberately distorted. Malicious nodes will send false information

to the fusion center in a centralized CRN. Malicious nodes could attempt to deny the service or to

cause other secondary nodes to violate spectrum allocation rules. To deny the service, a node will report

that free channels are used by the primary. To entice the neighbors to commit Federal Communication

Commission (FCC) violations, the occupancy report will show that channels used by the primary are

free. This attack strategy is called a secondary spectrum data falsification (SSDF), or Byzantine, attack.5

Thus, trust determination is a critical issue for CR networks.

Trust. The actual meaning of trust is domain and context specific. Consider, for example, networking;

at the MAC layer the multiple-access protocols assume that all senders follow the channel access

policy, e.g., in Carrier SenseMultiple Access with Collision Detection (CSMA-CD) a sender senses the

channel and then attempts to transmit if no one else does. In a store-and-forward network, trust assumes

that all routers follow a best-effort policy to forward packets toward their destination.

In the context of cognitive radio, trust is based on the quality of information regarding the channel

activity provided by a node. The status of individual channels can be assessed by each node based on

the results of its own measurements, combined with the information provided by its neighbors, as is the

case of several algorithms discussed in the literature [68,339].

The alternative discussed in Section 11.11 is to have a cloud-based service that collects information

from individual nodes, evaluates the state of each channel based on the information received, and

supplies this information on demand. Evaluation of the trust and identification of untrustworthy nodes

are critical for both strategies [284].

A Distributed Algorithm for TrustManagement in Cognitive Radio. The algorithm computes the

trust of node 1   i   n in each node in its vicinity, j ∈ Vi , and requires several preliminary steps. The

basic steps executed by a node i at time t are:

1. Determine node i ’s version of the occupancy report for each one of the K channels:

Si (t) = {si ,1(t), si ,2(t), . . . , si ,K (t)} (11.1)

In this step node i measures the power received on each of the K channels.

2. Determine the set Vi (t) of the nodes in the vicinity of node i . Node i broadcasts a message and

individual nodes in its vicinity respond with their NodeId.

3. Determine the distance to each node j ∈ Vi (t) using the algorithm described in this section.

4. Infer the power as measured by each node j ∈ Vi (t) on each channel k ∈ K.

5. Use the location and power information determined in the previous two steps to infer the status of

each channel:

sinfer

i ,k, j (t), 1   k   K, j ∈ Vi (t). (11.2)

A secondary node j should have determined 0 if the channel is free for use, 1 if the primary node is

active, and X if it cannot be determined.

sinfer

i ,k, j (t) =

⎧⎨

⎩

0 if secondary node j decides that channel k is free.

1 if secondary node j decides that channel k is used by the primary.

X if no inference can be made.

(11.3)

6. Receive the information provided by neighbor j ∈ Vi (t), Srecv

i ,k, j (t).

7. Compare the information provided by neighbor j ∈ Vi (t):

Srecv

i ,k, j (t) =

srecv

i ,1, j (t), srecv

i ,2, j (t), . . . , srecv

i ,K, j (t)

(11.4)

with the information inferred by node i about node j :

Sinfer

i ,k, j (t) =

sinfer

i ,1, j (t), sinfer

i ,2, j (t), . . . , sinfer

i ,K, j (t)

. (11.5)

8. Compute the number of matches, mismatches, and cases when no inference is possible, respectively,

αi , j (t) =M

Sinfer

i ,k, j (t), Srecv

i ,k, j (t)

, (11.6)

withMthe number of matches between the two vectors and

βi , j (t) = N

Sinfer

i ,k, j (t), Srecv

i ,k, j (t)

, (11.7)

with N the number of mismatches between the two vectors, and Xi , j (t) the number of cases where

no inference could be made.

9. Use the quantities αi , j (t), βi , j (t), and Xi , j (t) to assess the trust in node j . For example, compute

the trust of node i in node j at time t as

ζi , j (t) =

1 + Xi , j (t)

αi , j (t)

αi , j (t) + βi , j (t)

. (11.8)

Simulation of the Distributed Trust Algorithm. The cloud application is a simulation of a CRN to

assess the effectiveness of a particular trust assessment algorithm. Multiple instances of the algorithm

run concurrently on an AWS cloud. The area where the secondary nodes are located is partitioned into

several overlapping subareas, as shown in Figure 11.11. The secondary nodes are identified by an

instance Id, iId, as well as a global Id, gId. The simulation assumes that the primary nodes cover the

entire area; thus, their position is immaterial.

The simulation involves a controller and several cloud instances. In its initial implementation, the

controller runs on a local system under Linux Ubuntu 10.04 LTS. The controller supplies the data, the

trust program, and the scripts to the cloud instances; the cloud instances run under the Basic 32-bit Linux

image on AWS, the so-called t1.micro. The instances run the actual trust program and compute the

instantaneous trust inferred by a neighbor; the results are then processed by an awk6 script to compute

the average trust associated with a node as seen by all its neighbors. On the next version of the application

the data is stored on the cloud using the S3 service, and the controller also runs on the cloud.

In the simulation discussed here, the nodes with

gId = {1, 3, 6, 8, 12, 16, 17, 28, 29, 32, 35, 38, 39, 43, 44, 45} (11.9)

were programmed to be dishonest. The results show that the nodes programmed to act maliciously

have a trust value lower than that of the honest nodes; their trust value is always lower than 0.6 and,

in many instances, lower than 0.5 (see Figure 11.12). We also observe that the node density affects the

accuracy of the algorithm; the algorithm predicts more accurately the trust in densely populated areas.

As expected, nodes with no neighbors are unable to compute the trust.

In practice the node density is likely to be nonuniform, high in a crowded area such as a shopping mall,

and considerably lower in surrounding areas. This indicates that when the trust is computed using the

information provided by all secondary nodes, we can expect higher accuracy of the trust determination

in higher density areas.