Name: Vishal Rajesh Mahajan WT Home Assignment
Class: TE IT A Roll No: 56

Wireless Technology Home Assignment 1

Question 1

Question: We consider a cellular system in which total available voice channels to handle the traffic are 960.The area of each cell is 6 km2 and the total coverage area of the system is 2000 km2. Calculate (a) the system capacity if the cluster size, N (reuse factor), is 4 and (b) the system capacity if the cluster size is 7. How many times would a cluster of size 4 have to be replicated to cover the entire cellular area? Does decreasing the reuse factor N increase the system capacity? Explain.Use Python code to evaluate the answers and comment on the system capacity.

Python Code:

```
C_total = 960  # Total available voice channels
cell_area = 6  # Area of each cell in km^2
total_area = 2000  # Total coverage area in km^2

def calculate_capacity(N):
    cluster_area = N * cell_area  # Area of one cluster
    num_clusters = round(total_area / cluster_area)  # Proper rounding
    channels_per_cell = C_total / N  # Channels per cell
    system_capacity = num_clusters * C_total  # Total system capacity
    return cluster_area, num_clusters, channels_per_cell, system_capacity

cluster_area_N4, clusters_N4, channels_per_cell_N4, capacity_N4 = calculate_capacity(4)
cluster_area_N7, clusters_N7, channels_per_cell_N7, capacity_N7 = calculate_capacity(7)

# Output results
print("• Total available channels = 960")
print(f"• Cell area = {cell_area} km^2")
print(f"• Total coverage area = {total_area} km^2\n")

# For N = 4
print(f"• Area of a cluster with reuse N=4: {cluster_area_N4} km^2")
print(f"• Number of clusters for covering total area with N=4: {clusters_N4}")
print(f"• Number of channels per cell = {int(channels_per_cell_N4)}")
print(f"• System capacity = {clusters_N4} x {C_total} = {capacity_N4} channels\n")

# For N = 7
print(f"• Area of a cluster with reuse N=7: {cluster_area_N7} km^2")
print(f"• Number of clusters for covering total area with N=7: {clusters_N7}")
print(f"• Number of channels per cell = {int(channels_per_cell_N7)}")
print(f"• System capacity = {clusters_N7} x {C_total} = {capacity_N7} channels\n")

# Impact of decreasing N
if capacity_N4 > capacity_N7:
    print("Decreasing the reuse factor N increases the system capacity but increases interference.")
else:
    print("Decreasing the reuse factor N does not significantly increase the system capacity.")
```

Output:

```
• Total available channels = 960
• Cell area = 6 km^2
• Total coverage area = 2000 km^2

• Area of a cluster with reuse N=4: 24 km^2
• Number of clusters for covering total area with N=4: 83
• Number of channels per cell = 240
• System capacity = 83 x 960 = 79680 channels

• Area of a cluster with reuse N=7: 42 km^2
• Number of clusters for covering total area with N=7: 48
• Number of channels per cell = 137
• System capacity = 48 x 960 = 46080 channels

Decreasing the reuse factor N increases the system capacity but increases
interference.
```

It is evident when we decrease the value of N from 7 to 4, we increase the system capacity from 46,080 to 79,680 channels. Thus, decreasing the reuse factor (N) increases the system capacity.

Question 2

Question: Consider a cellular system with 395 total allocated voice channel frequencies. If the traffic is uniform with an average call holding time of 120 seconds and the call blocking during the system busy hour is 2%, calculate: 1. The number of calls per cell site per hour (i.e., call capacity of cell) 2. Mean S/I ratio for cell reuse factor equal to 4, 7, and 12. Assume omnidirectional antennas with six interferers in the first tier and a slope for path loss of 40 dB/decade ($\gamma = 4$).
Use Python code to evaluate the answers and comment on the system capacity.

Python Code:

```
import math
def erlang_B(traffic, m):
    B = 1.0
    for i in range(1, m + 1):
        B = (traffic * B) / (i + traffic * B)
    return B

def find_offered_traffic(m, target_B, tol=1e-6,
max_iter=1000):
    low = 0.0
    high = m * 2.0
    while erlang_B(high, m) < target_B:
        high *= 2.0

    iter_count = 0
    while (high - low) > tol and iter_count <
max_iter:
        mid = (low + high) / 2.0
        B_mid = erlang_B(mid, m)
        if B_mid < target_B:
            low = mid
        else:
            high = mid
        iter_count += 1

    return (low + high) / 2

def calc_S_over_I(N, gamma=4):
    ratio_linear = ((math.sqrt(3 * N)) ** gamma) /
6.0
    ratio_dB = 10 * math.log10(ratio_linear)
    return ratio_linear, ratio_dB

def main():
    total_channels = 395
    call_duration_sec = 120
    call_duration_hr = call_duration_sec / 3600.0
    blocking_probability = 0.02
    reuse_factors = [4, 7, 12]

    print("Cellular System Analysis \n")
    for N in reuse_factors:
        print(f"\nFor Reuse Factor N = {N}:")
        channels_per_cell = total_channels // N
        print(f" - Channels per cell: {channels_per_cell}")

        offered_traffic =
find_offered_traffic(channels_per_cell,
blocking_probability)
        carried_traffic = offered_traffic * (1 -
blocking_probability)  # Carried traffic
        call_capacity_per_cell = carried_traffic /
call_duration_hr
        print(f" - Offered Traffic: {offered_traffic:.2f}
Erlangs")
        print(f" - Carried Traffic: {carried_traffic:.2f}
Erlangs")
        print(f" - Call Capacity per Cell:
{call_capacity_per_cell:.2f} calls/hour")

        S_I_linear, S_I_dB = calc_S_over_I(N)
        print(f" - Mean S/I Ratio: {S_I_linear:.2f} (linear)")
        print(f" - Mean S/I Ratio: {S_I_dB:.2f} dB")

if __name__ == "__main__":
    main()
```

Output:

```
For Reuse Factor N = 4:
 - Channels per cell: 98
 - Offered Traffic: 86.04 Erlangs
 - Carried Traffic: 84.31 Erlangs
 - Call Capacity per Cell: 2529.44 calls/hour
 - Mean S/I Ratio: 24.00 (linear)
 - Mean S/I Ratio: 13.80 dB

For Reuse Factor N = 7:
 - Channels per cell: 56
 - Offered Traffic: 45.88 Erlangs
 - Carried Traffic: 44.96 Erlangs
 - Call Capacity per Cell: 1348.74 calls/hour
 - Mean S/I Ratio: 73.50 (linear)
 - Mean S/I Ratio: 18.66 dB

For Reuse Factor N = 12:
 - Channels per cell: 32
 - Offered Traffic: 23.72 Erlangs
 - Carried Traffic: 23.25 Erlangs
 - Call Capacity per Cell: 697.51 calls/hour
 - Mean S/I Ratio: 216.00 (linear)
 - Mean S/I Ratio: 23.34 dB
```

It is evident from the results that, by increasing the reuse factor from N = 4 to N = 12, the mean S/I ratio is improved from 13.8 to 23.3 dB. However, the call capacity of cell (i.e., calls per hour per cell) is reduced from 2529 to 697 calls per hour.

Question 3

Question: A QPSK/DSSS WLAN is designed to transmit in the 902–928 MHz ISM band. The symbol transmission rate is 0.5 Mega symbols/sec. An orthogonal code with 16 symbols is used. A bit error rate of 10 raised to -5 is required. How many users can be supported by the WLAN? A sector antenna with a gain of 2.6 is used. Assume interference factor β= 0.5 to account for the interference from users in other cells and power control efficiency α= 0.9. What is the bandwidth efficiency?

Python Code:

```
import math

f_high = 928
f_low = 902
Bw = f_high - f_low

Rs = 0.5e6
code_symbols = 16
bit_error_rate = 1e-5
antenna_gain = 2.6
beta = 0.5
alpha = 0.9

Gp = 26 / 2

Eb_N0_dB = 10
Eb_N0 = 10 ** (Eb_N0_dB / 10)

M = (Gp / Eb_N0) * (1 / (1 + beta))
* alpha * antenna_gain

Rb = Rs * math.log2(code_symbols)

total_data_rate = M * Rb
eta = total_data_rate / (Bw * 1e6)

print(f"Bandwidth (Bw): {Bw} MHz")
print(f"Bit rate per user (Rb): {Rb / 1e6:.2f} Mbps")
print(f"Number of users supported (M): {round(M)}")
print(f"Bandwidth efficiency (η): {eta:.2f} bps/Hz")

if eta < 0.5:
    capacity_comment = "The system has a low bandwidth
efficiency. Increasing the number of users or improving
coding techniques could enhance the capacity."
else:
    capacity_comment = "The system has a good bandwidth
efficiency and can support multiple users effectively."

print("\nSystem Capacity Analysis:")
print(capacity_comment)
```

Output:

```
Bandwidth (Bw): 26 MHz
Bit rate per user (Rb): 2.00 Mbps
Number of users supported (M): 2
Bandwidth efficiency (η): 0.16 bps/Hz

System Capacity Analysis:
The system has a low bandwidth efficiency. Increasing the number of users
 or improving coding techniques could enhance the capacity.
```

Question 4

Question: Consider a sender and receiver communicating over a 20 MHz wireless channel. a. The transmit power of the sender is 100 mW. Convert this value into units of dBm. b. The thermal noise on this channel is measured to be -100 dBm. Convert this value into units of watts. c. What is the SNR at the receiver if the transmitted signal suffers a path loss of 90 dB due to channel propagation effects? (Use the transmit power and noise values from (a) and (b) above. d. What is the maximum rate at which information can be transmitted between this sender and receiver as per Shannon's capacity formula? e. What are the answers to (c) and (d) above if the sender doubles his transmit power?

Python Code:

```
import math

def mw_to_dbm(mw):
    return 10 * math.log10(mw)

def dbm_to_watts(dbm):
    return 10 ** ((dbm - 30) / 10)

def calculate_snr(received_dbm, noise_dbm):
    snr_db = received_dbm - noise_dbm
    snr_linear = 10 ** (snr_db / 10)
    return snr_db, snr_linear

def shannon_capacity(bandwidth_hz, snr_linear):
    return bandwidth_hz * math.log2(1 +
snr_linear)

def main():
    power_mw = 100
    power_dbm = mw_to_dbm(power_mw)
    print("(a) 100 mW in dBm: {:.2f}
dBm".format(power_dbm))

    noise_dbm = -100
    noise_watts = dbm_to_watts(noise_dbm)
    print("(b) -100 dBm in Watts: {:.2e}
W".format(noise_watts))
received_power_dbm = 20 - 90
    snr_db, snr_linear = calculate_snr(received_power_dbm,
noise_dbm)
    print("(c) Received power: {:.2f}
dBm".format(received_power_dbm))
    print("    SNR: {:.2f} dB ({:.0f} in linear scale)".format(snr_db,
snr_linear))

    bandwidth = 20e6
    capacity = shannon_capacity(bandwidth, snr_linear)
    print("(d) Shannon capacity: {:.2f} Mbps".format(capacity / 1e6))

    new_transmit_dbm = mw_to_dbm(200)
    new_received_dbm = new_transmit_dbm - 90
    new_snr_db, new_snr_linear = calculate_snr(new_received_dbm,
noise_dbm)
    new_capacity = shannon_capacity(bandwidth, new_snr_linear)
    print("(e) With 200 mW transmit power:")
    print("    New transmit power: {:.2f}
dBm".format(new_transmit_dbm))
    print("    New received power: {:.2f}
dBm".format(new_received_dbm))
    print("    New SNR: {:.2f} dB ({:.0f} in linear
scale)".format(new_snr_db, new_snr_linear))
    print("    New Shannon capacity: {:.2f}
Mbps".format(new_capacity / 1e6))

if __name__ == "__main__":
    main()
```

Output:

```
(a) 100 mW in dBm: 20.00 dBm
(b) -100 dBm in Watts: 1.00e-13 W
(c) Received power: -70.00 dBm
    SNR: 30.00 dB (1000 in linear scale)
(d) Shannon capacity: 199.34 Mbps
(e) With 200 mW transmit power:
    New transmit power: 23.01 dBm
    New received power: -66.99 dBm
    New SNR: 33.01 dB (2000 in linear scale)
    New Shannon capacity: 219.33 Mbps
```