Northeastern University

EECE 7205 - FUNDAMENTALS OF COMPUTER ENGINEERING PROJECT 2

Bhuvan Karthik Channagiri

NU ID: 002825947

Contents

PART1 - PROGRAM SETUP

First, I have specified the Header Files and I have used Namespace as well for easier steps further in my code.

```
#include<iostream>
#include<vector>
#include<limits.h>
#include"Matrix.h"
#include<ctime>

using namespace std;
using namespace Numeric_lib;
```

```
struct task {
    int num; // the number of the task
    bool ct;  // judge whether the task is a cloud task
    double pri=0; //
    int FTl;  // the finish time of the task in a core
    int FTWS; // the finish time of the task in sending
    int FTC;   // the finish time of the task in cloud
    int FTWR;   // the finish time of the task in receiving
    int RTl; // the earlist  time that the task can start in local core
    int RTWS; // the earlist  time that the task can start in sending
    int RTC;  // the earlist  time that the task can start in cloud
    int RTWR; //the earlist  time that the task can start in receiving
    int ST; // the task's actual start time
    int chan; // illustrate which channel the task operate (local core = 0,1,2, cloud=3)
    bool exit;  //whether it is an exit task
    bool entry;  // whether it is an entry task
    int ready1;
    int ready2;
};
```

THEN, I DEFINE A NEW STRUCTURE DEFINED AS “TASK”.

Here I explain the function of each parameter in the new data structure “TASK”:

num: the serial number of the task

ct: judge whether the task is a cloud task

pri: the priority level of the task

FTL: if local schedule, the finish time of the task in local core

FTWS: if cloud schedule, the finish sending time of the task in wireless channel

FTC: if cloud schedule, the finish compute time of the task in cloud

FTWR: if cloud schedule, the finish receiving time of the task in wireless channel

RTL: if local schedule, the ready time of the task in local core

RTWS: if cloud schedule, the ready sending time of the task in wireless channel

RTC: if cloud schedule, the ready compute time of the task in cloud

RTWR: if cloud schedule, the ready receiving time of the task in wireless channel

ST: the actual start time of the task

chan: the location of the task(if 0,1,2, is in local core 1,2,3; if 3, is in cloud)

exit: judge whether the task is an exit task

entry: judge whether the task is an entry task

ready1: in kernel algorithm, the number of immediate predecessors of task which have not been scheduled

ready2: in kernel algorithm, the number of tasks in the same channel before the task which have not been scheduled

Here is the Phase one in step one: Primary Assignment.
In this first phase we will decide whether the task is to be executed on local or cloud.

```
void primary(vector<task>&ini, Matrix<int, 2>&ta,int t)
{
    int min;
    unsigned int i;
    unsigned int j;
    for (i = 0; i < ta.dim1(); i++)
    {
        ini[i].num = i + 1;
        min = ta(i, 0);
        for (j = 0; j < ta.dim2(); j++)
            if (ta(i, j) < min)
                min = ta(i, j);
        if (min > t)
            ini[i].ct = 1;
        else
            ini[i].ct = 0;
    }
}
```

HERE IS THE PHASE TWO IN STEP ONE: TASK PRIORITIZING. (I ALSO DETERMINE THE VALUE OF EXIT AND ENTRY)

```
void prioritize(vector<task>&ini, Matrix<int, 2>&ta, Matrix<int, 2>&G,int t)
{
    unsigned int i;
    unsigned int j,m;
    int k ;
    double w;
    double max;
    m = ini.size() - 1;
    for (i = 0; i < ini.size(); i++)
    {
        k = 0;
        for (j = 0; j < G.dim2(); j++)
            if (G(m - i, j) == 1)
                k = k + 1;
        if (k == 0)
            ini[m - i].exit = 1;
        k = 0;
        for (j = 0; j < G.dim2(); j++)
            if (G(j, m - i) == 1)
                k = k + 1;
        if(k==0)
            ini[m - i].entry = 1;
        max = 0;
        w = 0;
        if (!(ini[m - i].ct))
        {
            for (j = 0; j < ta.dim2(); j++)
                w = w + ta(m - i, j);
            w = w / 3;
        }
        else
            w = t;
        for (j = 0; j < G.dim2(); j++)
            if ((G(m - i, j) == 1) && (max < ini[j].pri))
                max = ini[j].pri;
        ini[m - i].pri = w + max;
    }
}
```

HERE IS THE MAIN FUNCTION OF PHASE THREE IN STEP ONE: EXECUTION UNIT SELECTION.
I USE “MINT” TO COMPUTE THE MINIMUM FINISH TIME IN LOCAL CORE AND USE “ANOT” TO COMPUTE THE MINIMUM FINISH TIME IN CLOUD. THEN COMPARE THIS TWO VALUES TO DETERMINE WHETHER THE TASK IN CLOUD OR LOCAL CORE.

```
void initials(vector<task>&S, vector<task>&ini, Matrix<int, 2>&ta, Matrix<int, 2>&G, int ts, int tc, int tr)
{
    unsigned int i;
    int t;
    int maxp; // find the max priority in each iteration of ini
    int mint;  // find the minimum finish time of local
    int anot;  // perpare for another time (cloud)
    t = ts + tc + tr;
    for (i = 0; i < G.dim1(); i++)
    {
        maxp = find_biggest_priority(ini);
        if (!ini[maxp].ct)
        {
            mint = locals(ini[maxp], S, G, ta);
            anot = clouds(ini[maxp], S, G, ts, tc, tr);
            if (anot < mint)
            {
                ini[maxp].RTl = 0;
                ini[maxp].FTl = 0;
                ini[maxp].chan = 3;
                ini[maxp].FTWR = anot;
                ini[maxp].ST = anot - t;
            }
            else
            {
                ini[maxp].FTC = 0;
                ini[maxp].FTWS = 0;
                ini[maxp].RTWS = 0;
                ini[maxp].RTC = 0;
                ini[maxp].RTWR = 0;
                ini[maxp].FTWR = 0;
                ini[maxp].FTl = mint;
                ini[maxp].ST = mint - ta(ini[maxp].num - 1, ini[maxp].chan);
            }
        }
        else
        {
            ini[maxp].FTl = 0;
            ini[maxp].RTl = 0;
            ini[maxp].chan = 3;
            ini[maxp].FTWR= clouds(ini[maxp], S, G, ts, tc, tr);
            ini[maxp].ST = ini[maxp].FTWR - t;
        }
        S.push_back(ini[maxp]);
        ini.erase(ini.begin() + maxp);
    }
}
```

HERE IS THE FUNCTION WHICH RETURNS THE MAXIMUM FINISH TIME OF TASK IN LOCAL CORE.

```
int locals(task &vi, vector<task>&S, Matrix<int, 2>&G, Matrix<int, 2>&ta)
{
    vi.RTl = d_rtl(vi, S, G);
    unsigned int i;
    unsigned int j;
    int mint=INT_MAX;
    int ft;
    int max = 0; // find a local core's biggest finish time
    if (S.size()==0)
    {
        for (i = 0; i < ta.dim2(); i++)
        {
            ft = ta(vi.num - 1, i);
            if (mint > ft)
            {
                mint = ft;
                vi.chan = i;
            }
        }
        return mint;
    }
    for (i = 0; i < ta.dim2(); i++)
    {
        ft = vi.RTl + ta(vi.num - 1, i);
        max = 0;
        for (j = 0; j < S.size(); j++)
            if ((S[j].chan == i) && (max < S[j].FTl))
                max = S[j].FTl;
        if(max>vi.RTl)
            ft=max+ ta(vi.num - 1, i);
        if (mint > ft)
        {
            mint = ft;
            vi.chan = i;
        }
    }
    return mint;
}
```

THE FIRST FUNCTION RETURNS THE SERIAL NUMBER OF THE TASK WHOSE PRIORITY IS BIGGEST.
THE THIRD FUNCTION RETURNS RTL OF THE TASK IF LOCAL SCHEDULE.

```
int find_biggest_priority(vector<task>&ini)
{
    unsigned int i;
    int max=0;
    for (i = 0; i < ini.size(); i++)
        if (ini[i].pri > ini[max].pri)
            max = i;
    return max;
}

// find the max in two numbers
Comment Code
int maximum(int &m, int &n)
{
    if (m >= n)
        return m;
    else
        return n;
}

// if local schedule, return RTL
Comment Code
int d_rtl(task &vi, vector<task>&S, Matrix<int, 2>&G)
{
    unsigned int i;
    unsigned int j;
    int max=0;
    if (S.size()!=0)
    {
        for (i = 0; i < G.dim2(); i++)
            if (G(i, vi.num - 1) == 1)
                for (j = 0; j < S.size(); j++)
                    if ((S[j].num == i + 1)&&(max < maximum(S[j].FTl, S[j].FTWR)))
                        max = maximum(S[j].FTl, S[j].FTWR);
    }
    return max;
}
```

HERE IS THE FUNCTION WHICH RETURNS THE FINISH RECEIVING TIME OF TASK IN WIRELESS CHANNEL IF CLOUD SCHEDULE. ALSO COMPUTE FTWS AND FTC.

```
int clouds(task &vi, vector<task>&S, Matrix<int, 2>&G, int ts, int tc, int tr)
{
    vi.RTWS = d_rtws(vi, S, G);
    unsigned int i;
    int maxs = 0;
    int t;
    int maxc = 0;
    int maxr = 0;
    int ft;
    t = ts + tc + tr;
    if (S.size()==0)
    {
        vi.FTWS = ts;
        vi.RTC = ts;
        vi.FTC = ts + tc;
        vi.RTWR = ts+tc;
        return t;
    }
    for(i=0;i<S.size();i++)
        if (S[i].chan == 3)
            if (maxs < S[i].FTWS)
                maxs = S[i].FTWS;
    if (maxs > vi.RTWS)
        vi.FTWS = maxs + ts;
    else
        vi.FTWS = vi.RTWS + ts;
    vi.RTC = d_rtc(vi, S, G);
    for (i = 0; i < S.size(); i++)
        if (S[i].chan == 3)
            if (maxc < S[i].FTC)
                maxc = S[i].FTC;
    if (maxc > vi.RTC)
        vi.FTC = maxc + tc;
    else
        vi.FTC = vi.RTC + tc;
    vi.RTWR = d_rtwr(vi);
    for (i = 0; i < S.size(); i++)
        if (S[i].chan == 3)
            if (maxr < S[i].FTWR)
                maxr = S[i].FTWR;
    if (maxr > vi.RTWR)
        ft = maxr + tr;
    else
        ft = vi.RTWR + tr;
    return ft;
}
```

HERE I COMPUTE RTWS, RTC, RTWR OF THE TASK IF CLOUD SCHEDULE.

```
// if its a Cloud schedule, return RTWS(Earliest start time for wireless sending)
Comment Code
int d_rtws(task &vi, vector<task>&S, Matrix<int, 2>&G)
{
    unsigned int i;
    unsigned int j;
    int max=0;
    if (S.size()!=0)
    {
        for (i = 0; i < G.dim2(); i++)
            if (G(i, vi.num - 1) == 1)
                for (j = 0; j < S.size(); j++)
                    if ((S[j].num == i + 1)&&(max < maximum(S[j].FTl, S[j].FTWS)))
                        max = maximum(S[j].FTl, S[j].FTWS);
    }
    return max;
}

// if its a Cloud schedule, return RTC(Earliest start time for wireless computing)
Comment Code
int d_rtc(task &vi, vector<task>&S, Matrix<int, 2>&G)
{
    unsigned int i;
    unsigned int j;
    int max=vi.FTWS;
    if (S.size()!=0)
    {
        for (i = 0; i < G.dim2(); i++)
            if (G(i, vi.num - 1) == 1)
                for (j = 0; j < S.size(); j++)
                    if ((S[j].num == i + 1)&&(max < maximum(vi.FTWS, S[j].FTC)))
                        max = maximum(vi.FTWS, S[j].FTC);
    }
    return max;
}

// if its a Cloud schedule, return RTWR(Earliest start time to start for wireless receiving)
Comment Code
int d_rtwr(task &vi)
{
    return vi.FTC;
}
```

THE FIRST FUNCTION RETURNS THE ACTUAL FINISH TIME OF THE TASK.
THE SECOND FUNCTION PRINTS THE SCHEDULE OF ALL TASKS (INVOLVING THE CHANNEL OF TASK, THE START AND FINISH TIME OF TASK)

```
int find_ft(task&vi)
{
    int max;
    max = maximum(vi.FTl, vi.FTWR);
    return max;
 }

// print the sequence S
Comment Code
void prints(vector<task>&S)
{
    unsigned int i;
    int k,m;
    for (i = 0; i < S.size(); i++)
    {
        k = 1 + S[i].chan;
        m = find_ft(S[i]);
        cout << "Task" << S[i].num << ": ";
        switch (S[i].chan)
        {
        case 0:
            cout << "local core" << k << ", ";
            break;
        case 1:
            cout << "local core" << k << ", ";
            break;
        case 2:
            cout << "local core" << k << ", ";
            break;
        case 3:
            cout << "cloud" << ", ";
            break;
        default:
            break;
        }
        cout << "start time is: " << S[i].ST << ",finish time is: "<<m<<endl;
    }
}
```

THE FIRST FUNCTION RETURNS THE TOTAL COMPLETION TIME OF THE SCHEDULE.
THE SECOND FUNCTION RETURNS THE TOTAL ENERGY CONSUMED IN THE SCHEDULE.

```
int find_tcom(vector<task>&S)
{
    unsigned int i;
    int max=0;
    for (i = 0; i < S.size(); i++)
        if ((S[i].exit) && (max < find_ft(S[i])))
            max = find_ft(S[i]);
    return max;
}

// return the total energy of the sequence S
Comment Code
double find_en(vector<task>&S, int p1, int p2, int p3, double ps)
{
    unsigned int i;
    double ene=0;
    for (i = 0; i < S.size(); i++)
    {
        switch (S[i].chan)
        {
        case 0:
            ene = ene + p1 * (find_ft(S[i]) - S[i].ST);
            break;
        case 1:
            ene = ene + p2 * (find_ft(S[i]) - S[i].ST);
            break;
        case 2:
            ene = ene + p3 * (find_ft(S[i]) - S[i].ST);
            break;
        case 3:
            ene = ene + ps * (S[i].FTWS - S[i].ST);
            break;
        default:
            break;
        }
    }
    return ene;
}
```

HERE IS STEP TWO: TASK MIGRATION.
THE OUTER LOOP REPEATS TO RUN THE “MCC” FUNCTION UNTIL THE ENERGY CONSUMED IS MINIMIZED.

```
void outerloop(vector<task>&S, Matrix<int, 2>&G, Matrix<int, 2>&ta, int ts, int tc, int tr, int p1, int p2, int p3, double ps, int tmax)
{
    double en;
    double en1=0;
    en = find_en(S, p1, p2, p3, ps);
    while (en1<en)
    {
        en= find_en(S, p1, p2, p3, ps);
        mcc(S, G, ta, ts, tc, tr, p1, p2, p3, ps, tmax);
        en1= find_en(S, p1, p2, p3, ps);
    }
}
```

HERE IS THE “MCC” FUNCTION WHICH INVOLVES KERNEL ALGORITHM.

```
void mcc(vector<task>&S, Matrix<int, 2>&G, Matrix<int, 2>&ta, int ts, int tc, int tr,int p1, int p2, int p3, double ps, int tmax)
{
    unsigned int i, j;
    int tcom;
    int tcom2;
    int a;
    double en;
    double en1;
    double en2;
    double ratio1=0;
    double ratio2;
    vector<task>SN;
    tcom = find_tcom(S);
    en = find_en(S, p1, p2, p3, ps);
    for (i = 0; i < S.size(); i++)
    {
        a = S[i].chan;
        if (S[i].chan != 3)
        {
            for (j = 0; j < 4; j++)
            {
                if (j != a)
                {
                    SN.erase(SN.begin(), SN.end());
                    en1 = find_en(S, p1, p2, p3, ps);
                    kernel(S, SN, j, S[i], G, ta, ts, tc, tr);
                    tcom2 = find_tcom(SN);
                    en2 = find_en(SN, p1, p2, p3, ps);
                    if ((en2 < en1) && (tcom >= tcom2))
                        S = SN;
                    else if ((en2 < en1) && (tcom2 <= tmax))
                    {
                        ratio2 = (en - en2) / (tcom2 - tcom);
                        if (ratio2 > ratio1)
                        {
                            ratio1 = ratio2;
                            S = SN;
                        }
                    }
                }
            }
        }
    }
}
```

HERE IS THE FUNCTION WHICH IS THE IMPLEMENTATION OF KERNEL ALGORITHM.

```
void kernel(vector<task>&S, vector<task>&SN, int ktar, task vtar, Matrix<int, 2>&G, Matrix<int, 2>&ta,int ts, int tc, int tr)
{
    unsigned int i;
    int m;
    int t;
    t = ts + tc + tr;
    vector<task>re;
    re = S;
    for (i = 0; i < re.size(); i++)
        if (vtar.num == re[i].num)
        {
            re[i].chan = ktar;
            if (ktar == 3)
            {
                re[i].FTl = 0;
                re[i].RTl = 0;
            }
        }
    while (re.size()!=0)
    {
        get_ready1(re, G);
        get_ready2(re);
        m = 0;
        while ((re[m].ready1!=0)&&(re[m].ready2 != 0))
            m = m + 1;
        if (re[m].chan == 3)
        {
            re[m].FTWR = clouds(re[m], SN, G, ts, tc, tr);
            re[m].ST = re[m].FTWR - t;
        }
        else
        {
            re[m].FTl = localse(re[m], SN, G, ta);
            re[m].ST = re[m].FTl - ta(re[m].num - 1, re[m].chan);
        }
        SN.push_back(re[m]);
        re.erase(re.begin() + m);
    }
}
```

THE FIRST FUNCTION IS TO COMPUTE THE VALUE OF READY1.
THE SECOND FUNCTION IS TO COMPUTE THE VALUE OF READY2.

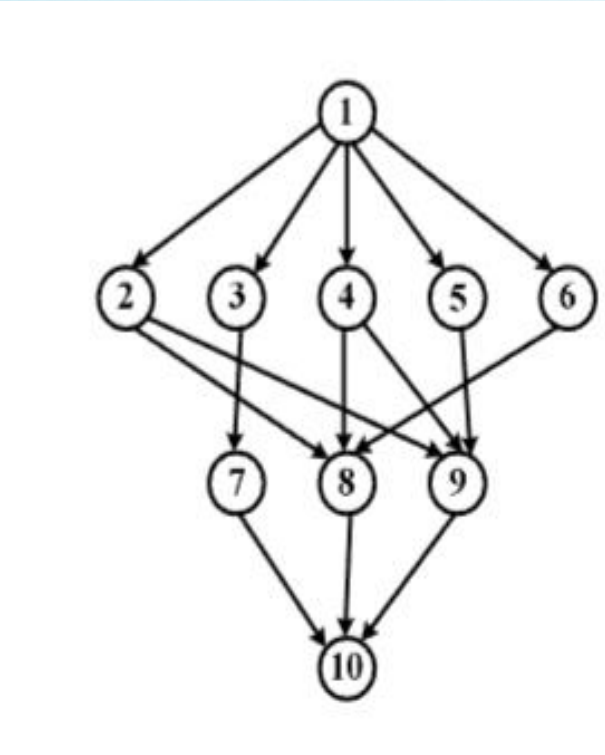
```
void get_ready1(vector<task>&S, Matrix<int, 2>&G)
{
    unsigned int i, j, k;
    int m;
    for (i = 0; i < S.size(); i++)
    {
        m = 0;
        for (j = 0; j < G.dim2(); j++)
            if (G(j, S[i].num-1) == 1)
                for (k = 0; k < S.size(); k++)
                    if (S[k].num == j + 1)
                        m = m + 1;
        S[i].ready1 = m;
    }
}

//compute all the ready2 in a sequence
Comment Code
void get_ready2(vector<task>&S)
{
    unsigned int i, j;
    int m;
    for (i = 0; i < S.size(); i++)
    {
        m = 0;
        for (j = 0; j < S.size(); j++)
            if ((S[i].chan == S[j].chan) && (S[j].ST < S[i].ST))
                m = m + 1;
        S[i].ready2 = m;
    }
}
```

PART - 2 INPUT AND OUTPUT

- THIS PART I WILL USE FIVE EXAMPLES TO VERIFY THE CORRECTNESS OF MY CODES.
- THE FIRST EXAMPLE IS THE SAME WITH PAPER [1].
- THE OTHER EXAMPLES ARE GOOD EXAMPLES TO SHOW HOW THE CLOUD CAN SAVE ENERGY FOR THE WHOLE SCHEDULE.

INPUT OF THE FIRST EXAMPLE



$$T_{max} = 27$$

$$\begin{cases} T^S = 3; \\ T^C = 1; \\ T^R = 1; \end{cases}$$

$$\begin{cases} P_1 = 1; \\ P_2 = 2; \\ P_3 = 4; \\ P_s = 0.5; \end{cases}$$

Time in each core	local core1	local core2	local core3
Task 1	9	7	5
Task 2	8	6	5
Task 3	6	5	4
Task 4	7	5	3
Task 5	5	4	2
Task 6	7	6	4
Task 7	8	5	3
Task 8	6	4	2
Task 9	5	3	2
Task 10	7	4	2

AFTER INITIAL SCHEDULE

THE COMPLETION TIME OF THE SCHEDULE IS 18.

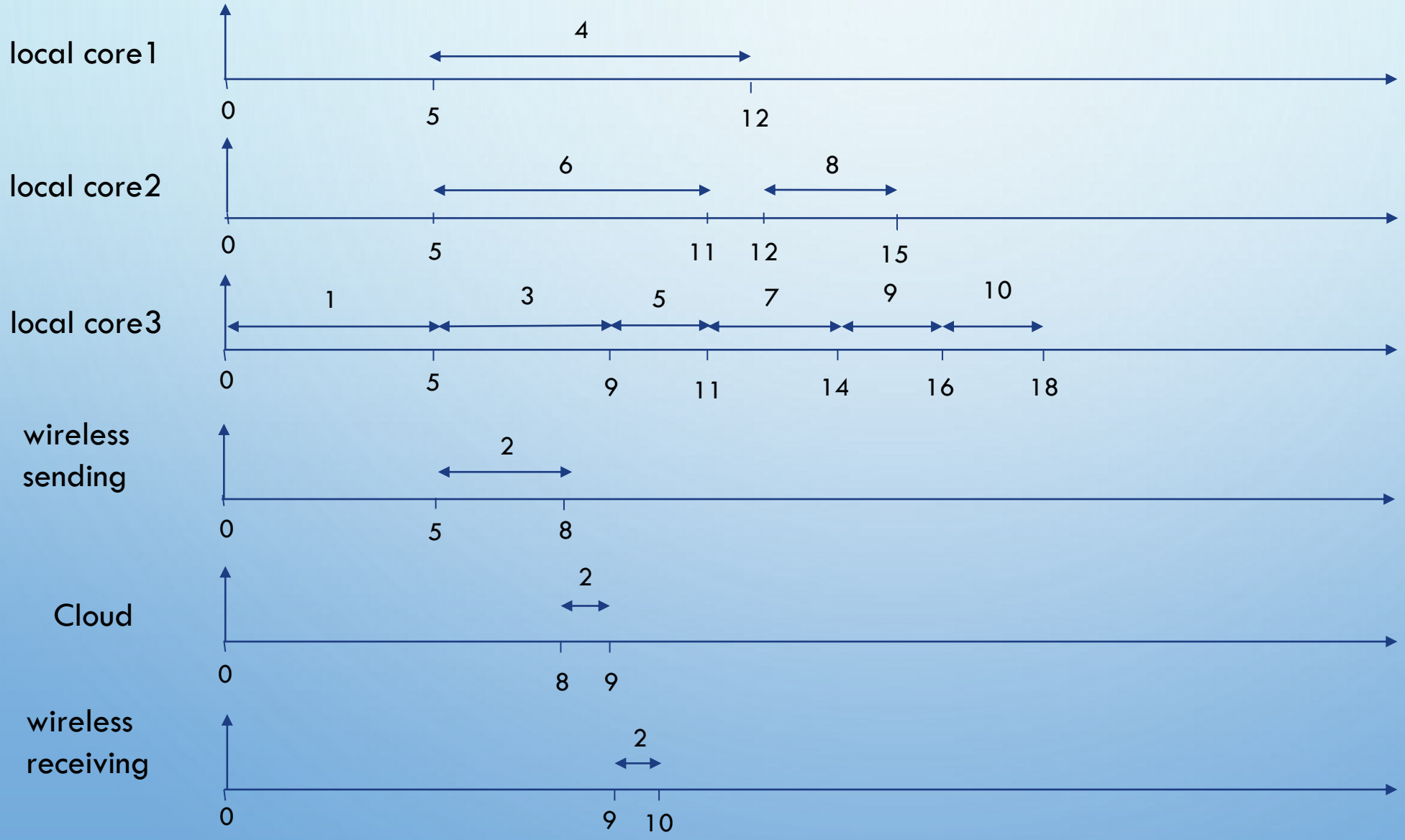
THE TOTAL ENERGY CONSUMED IS 100.5.

THE RUNNING TIME OF THE INITIAL SCHEDULE PROGRAM IS 0.019 MS.

```
Initial schedule:
Task1: local core3, start time is: 0,finish time is: 5
Task3: local core3, start time is: 5,finish time is: 9
Task2: cloud, start time is: 5,finish time is: 10
Task6: local core2, start time is: 5,finish time is: 11
Task4: local core1, start time is: 5,finish time is: 12
Task5: local core3, start time is: 9,finish time is: 11
Task7: local core3, start time is: 11,finish time is: 14
Task8: local core2, start time is: 11,finish time is: 15
Task9: local core3, start time is: 14,finish time is: 16
Task10: local core3, start time is: 16,finish time is: 18
 Now the total energy is: 100.5
 Now the completion time is: 18
```

E1=(7*1)=7; E2=(6+4)*2=20;E3=(5+4+2+3+2+2)*4=72;Es=(3*0.5)=1.5;
E=7+20+72+1.5=100.5

OUTPUT OF THE FIRST EXAMPLE AFTER INITIAL SCHEDULE



AFTER TASK MIGRATION

THE COMPLETION TIME OF THE SCHEDULE IS 27.

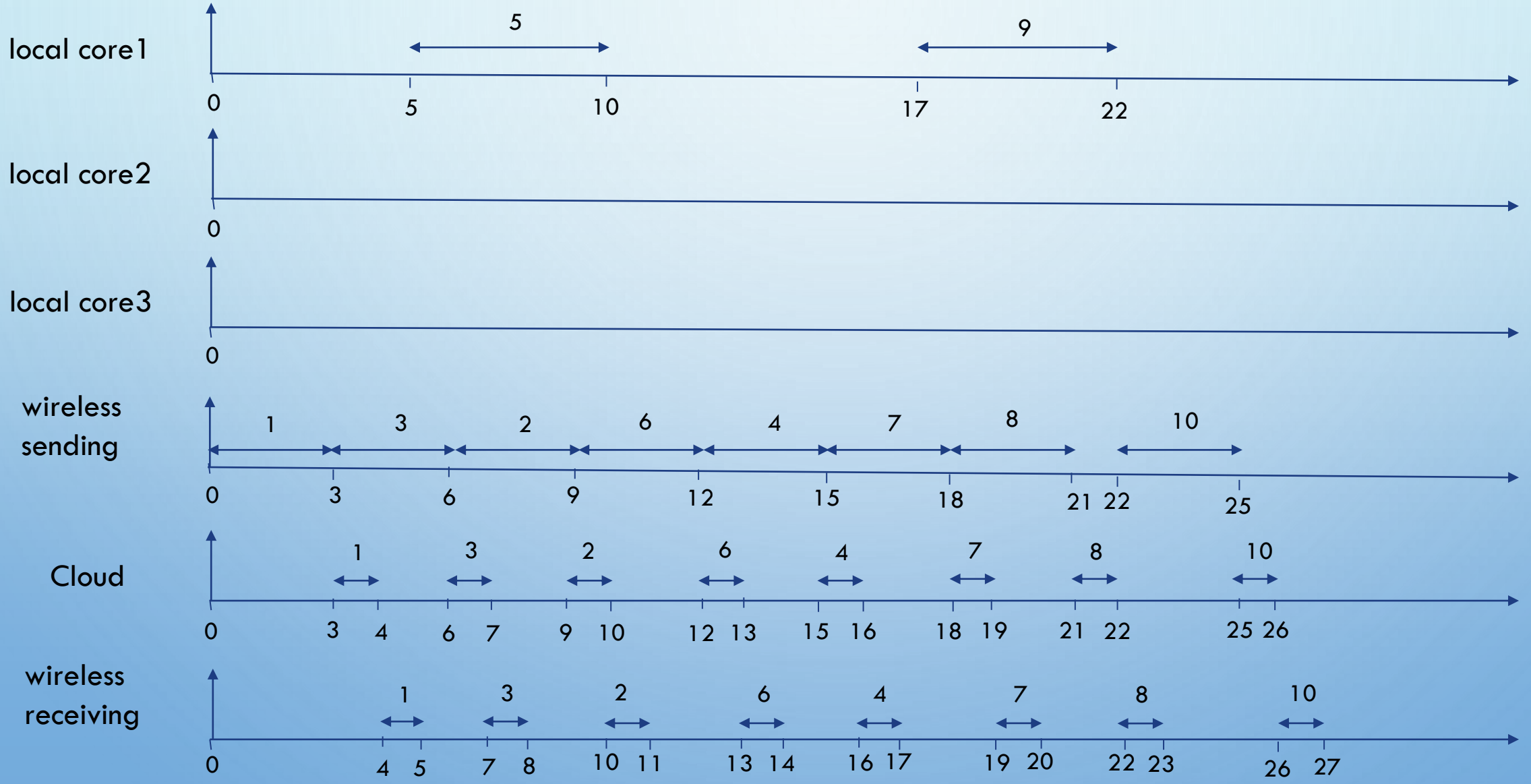
THE TOTAL ENERGY CONSUMED IS 22.

THE RUNNING TIME OF THE INITIAL SCHEDULE PROGRAM IS 0.983 ms.

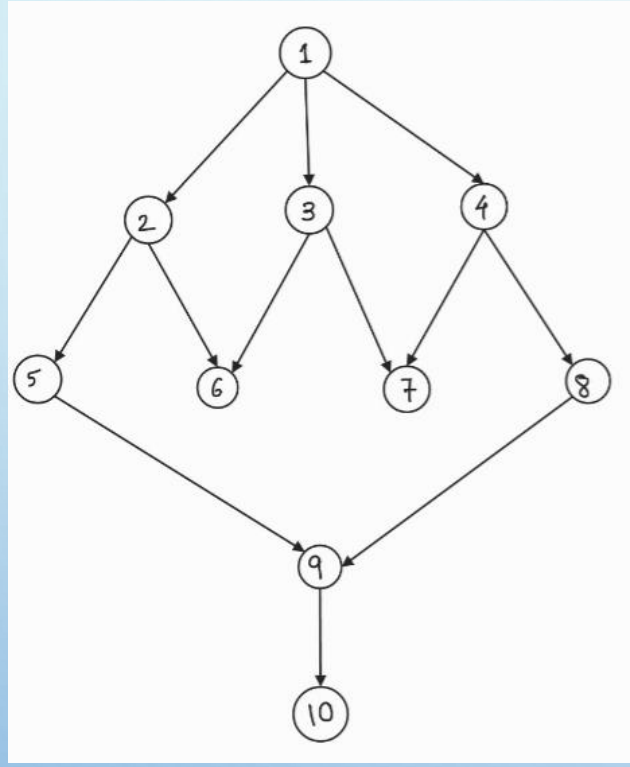
```
The running time of initial schedule of Graph is 0.023 ms
After Task Migration:
Task1: cloud, start time is: 0,finish time is: 5
Task2: cloud, start time is: 3,finish time is: 8
Task4: cloud, start time is: 6,finish time is: 11
Task6: cloud, start time is: 9,finish time is: 14
Task3: cloud, start time is: 12,finish time is: 17
Task5: local core1, start time is: 5,finish time is: 10
Task8: cloud, start time is: 15,finish time is: 20
Task7: cloud, start time is: 18,finish time is: 23
Task9: local core1, start time is: 17,finish time is: 22
Task10: cloud, start time is: 22,finish time is: 27
 Now the total energy is: 22
 Now the completion time is: 27
The running time of task migration of Graph 1 is 0.983 ms
```

E1=(5+5)*1=10; E2=0;E3=0;Es=(3*8*0.5)=12;
E=10+12=22

OUTPUT OF THE FIRST EXAMPLE AFTER TASK MIGRATION



INPUT OF THE SECOND EXAMPLE



Time in each core	local core1	local core2	local core3
Task 1	9	7	5
Task 2	8	6	5
Task 3	6	5	4
Task 4	7	5	3
Task 5	5	4	2
Task 6	7	6	4
Task 7	8	5	3
Task 8	6	4	2
Task 9	5	3	2
Task 10	7	4	2

$$T_{max} = 27$$

$$\begin{cases} T^S = 3; \\ T^C = 1; \\ T^R = 1; \end{cases}$$

$$\begin{cases} P_1 = 1; \\ P_2 = 2; \\ P_3 = 4; \\ P_s = 0.5; \end{cases}$$

INITIAL SCHEDULE

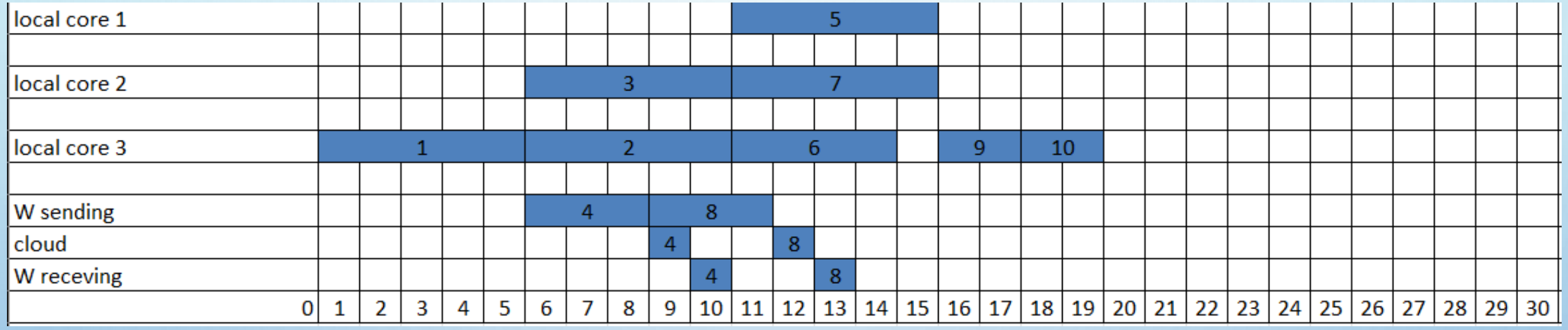
THE COMPLETION TIME OF THE SCHEDULE IS 19.

THE TOTAL ENERGY CONSUMED IS 100.

THE RUNNING TIME OF THE INITIAL SCHEDULE PROGRAM IS 0.025ms.

```
Initial schedule:
Task1: local core3, start time is: 0,finish time is: 5
Task2: local core3, start time is: 5,finish time is: 10
Task3: local core2, start time is: 5,finish time is: 10
Task4: cloud, start time is: 5,finish time is: 10
Task6: local core3, start time is: 10,finish time is: 14
Task7: local core2, start time is: 10,finish time is: 15
Task8: cloud, start time is: 8,finish time is: 13
Task5: local core1, start time is: 10,finish time is: 15
Task9: local core3, start time is: 15,finish time is: 17
Task10: local core3, start time is: 17,finish time is: 19
 Now the total energy is: 100
 Now the completion time is: 19
Running time of initial schedule of Graph is 0.025 ms
After Task Migration:
Task1: cloud, start time is: 0,finish time is: 5
Task2: cloud, start time is: 3,finish time is: 8
Task3: cloud, start time is: 6,finish time is: 11
Task4: cloud, start time is: 9,finish time is: 14
Task6: cloud, start time is: 12,finish time is: 17
Task7: cloud, start time is: 15,finish time is: 20
Task8: cloud, start time is: 18,finish time is: 23
Task5: local core1, start time is: 8,finish time is: 13
Task9: local core3, start time is: 23,finish time is: 25
Task10: local core3, start time is: 25,finish time is: 27
 Now the total energy is: 31.5
 Now the completion time is: 27
Running time of task migration of Graph is 1.596 ms
```

OUTPUT OF THE SECOND EXAMPLE AFTER INITIAL SCHEDULE



E1=5*1=5;

E2 = (5 + 5)*2 = 20;

E3 = (5 + 5 + 4 +2+2)*4=72;

E_S = (3 + 3)*0.5 = 3;

E = 5+10+72+3 =100

AFTER TASK MIGRATION

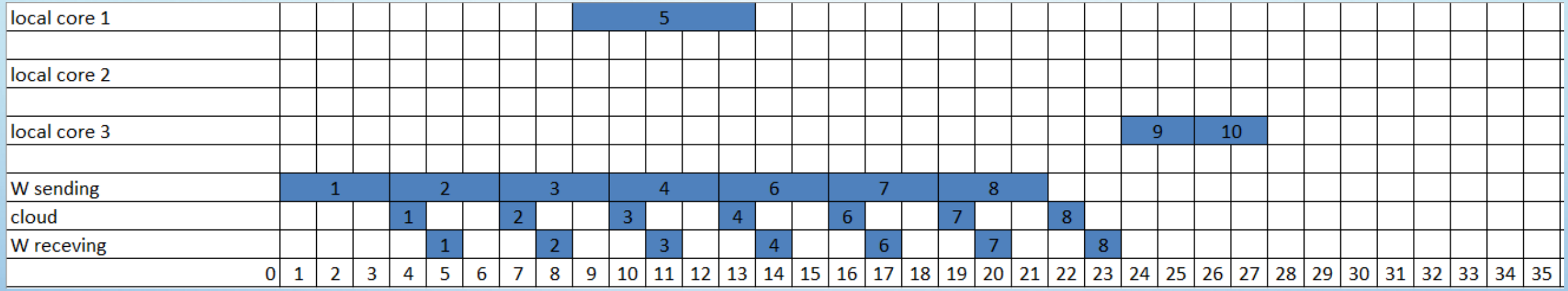
THE COMPLETION TIME OF THE SCHEDULE IS 27.

THE TOTAL ENERGY CONSUMED IS 31.5.

THE RUNNING TIME OF THE INITIAL SCHEDULE PROGRAM IS 1.596 ms.

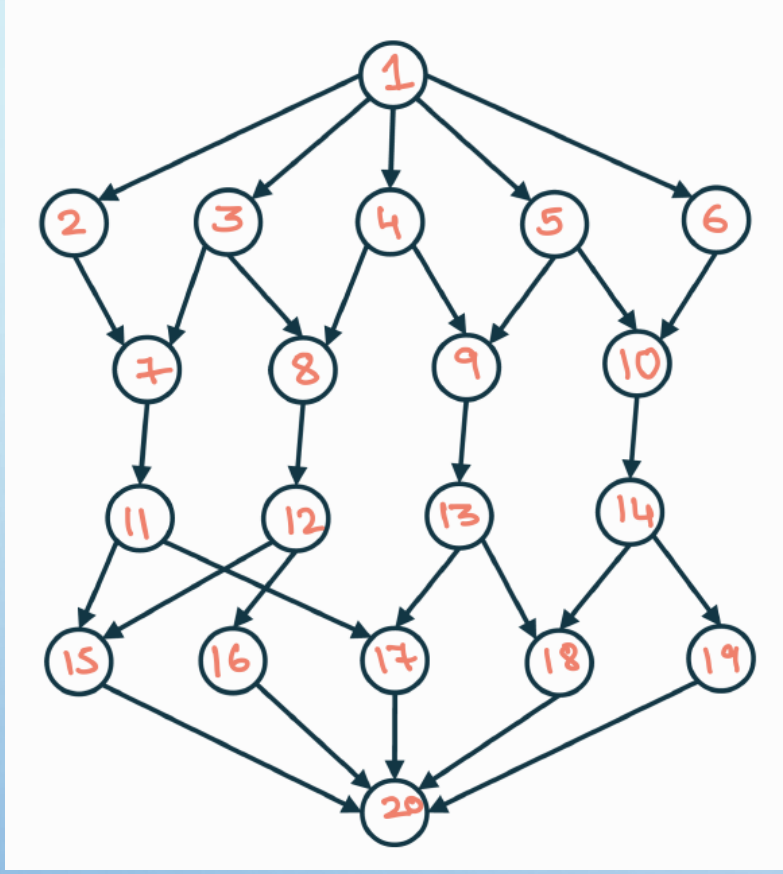
```
Initial schedule:
Task1: local core3, start time is: 0,finish time is: 5
Task2: local core3, start time is: 5,finish time is: 10
Task3: local core2, start time is: 5,finish time is: 10
Task4: cloud, start time is: 5,finish time is: 10
Task6: local core3, start time is: 10,finish time is: 14
Task7: local core2, start time is: 10,finish time is: 15
Task8: cloud, start time is: 8,finish time is: 13
Task5: local core1, start time is: 10,finish time is: 15
Task9: local core3, start time is: 15,finish time is: 17
Task10: local core3, start time is: 17,finish time is: 19
 Now the total energy is: 100
 Now the completion time is: 19
Running time of initial schedule of Graph is 0.025 ms
After Task Migration:
Task1: cloud, start time is: 0,finish time is: 5
Task2: cloud, start time is: 3,finish time is: 8
Task3: cloud, start time is: 6,finish time is: 11
Task4: cloud, start time is: 9,finish time is: 14
Task6: cloud, start time is: 12,finish time is: 17
Task7: cloud, start time is: 15,finish time is: 20
Task8: cloud, start time is: 18,finish time is: 23
Task5: local core1, start time is: 8,finish time is: 13
Task9: local core3, start time is: 23,finish time is: 25
Task10: local core3, start time is: 25,finish time is: 27
 Now the total energy is: 31.5
 Now the completion time is: 27
Running time of task migration of Graph is 1.596 ms
```

OUTPUT OF THE SECOND EXAMPLE AFTER TASK MIGRATION



E_S=(3*7)*0.5 = 10.5;
E1=5*1=5;
E3=4*4=16;
E = 10.5+5+16 = 31.5;

INPUT OF THIRD EXAMPLE



Tmax = 40

$$\begin{cases} T^S = 3; \\ T^C = 1; \\ T^R = 1; \end{cases} \qquad \begin{cases} P_1 = 1; \\ P_2 = 2; \\ P_3 = 4; \\ P_s = 0.5; \end{cases}$$

Time in each core	local core1	local core2	local core3
Task 1	9	7	5
Task 2	8	6	5
Task 3	6	5	4
Task 4	7	5	3
Task 5	5	4	2
Task 6	7	6	4
Task 7	8	5	3
Task 8	6	4	2
Task 9	5	3	2
Task 10	7	4	2
Task 11	7	4	2
Task 12	6	3	2
Task 13	5	3	3
Task 14	8	7	4
Task 15	9	8	5
Task 16	6	4	2
Task 17	7	5	3
Task 18	8	4	2
Task 19	9	6	3
Task 20	7	6	4

INITIAL SCHEDULE

THE COMPLETION TIME OF THE SCHEDULE IS 30.
THE TOTAL ENERGY CONSUMED IS 180.5.
THE RUNNING TIME OF THE INITIAL SCHEDULE PROGRAM IS 0.061 ms.

```
Initial schedule:
Task1: local core3, start time is: 0,finish time is: 5
Task2: local core3, start time is: 5,finish time is: 10
Task6: cloud, start time is: 5,finish time is: 10
Task3: local core2, start time is: 5,finish time is: 10
Task5: local core1, start time is: 5,finish time is: 10
Task4: local core3, start time is: 10,finish time is: 13
Task7: local core2, start time is: 10,finish time is: 15
Task10: local core3, start time is: 13,finish time is: 15
Task8: local core3, start time is: 15,finish time is: 17
Task14: cloud, start time is: 15,finish time is: 20
Task9: local core1, start time is: 13,finish time is: 18
Task11: local core2, start time is: 15,finish time is: 17
Task12: local core3, start time is: 17,finish time is: 19
Task13: local core2, start time is: 18,finish time is: 21
Task15: local core3, start time is: 19,finish time is: 24
Task19: cloud, start time is: 18,finish time is: 23
Task17: local core2, start time is: 21,finish time is: 26
Task18: local core3, start time is: 24,finish time is: 26
Task16: local core1, start time is: 19,finish time is: 25
Task20: local core3, start time is: 26,finish time is: 30
 Now the total energy is: 180.5
 Now the completion time is: 30
The running time of initial schedule of Graph is 0.061 ms
```

OUTPUT OF THE THIRD EXAMPLE AFTER INITIAL SCHEDULE

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
LOCAL CORE 1						5								9						16										
LOCAL CORE 2						3					7					11		13				17								
LOCAL CORE 3	1					2					4			10		8		12		15					18			20		
WIRELESS SENDING						6										14			19										
CLOUD									6										14			19							
WIRELESS RECEIVING										6										14			19						

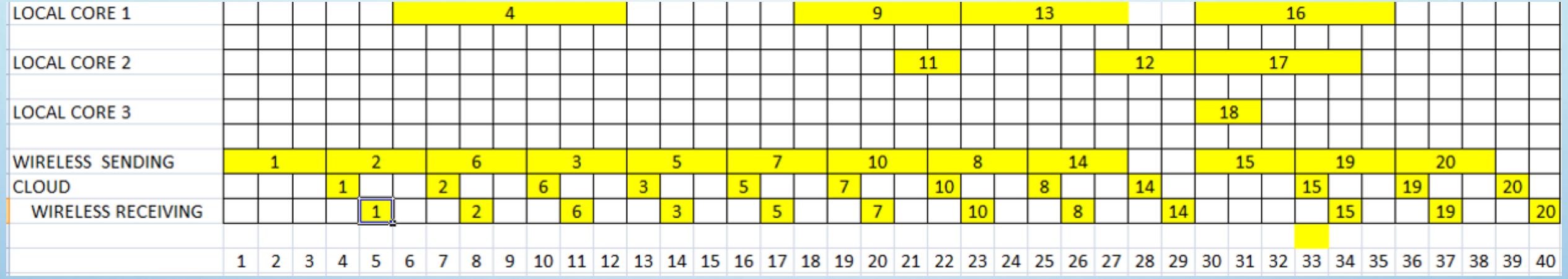
E1=5+5+6=16;
E2=(20)*2=40;
E3=30*4=120;
Es=(3*3*0.5)=4.5;
E=16+10+120+4.5=180.5

TASK MIGRATION

THE COMPLETION TIME OF THE SCHEDULE IS 40.
THE TOTAL ENERGY CONSUMED IS 69.
THE RUNNING TIME OF THE INITIAL SCHEDULE PROGRAM IS 25.615 ms.

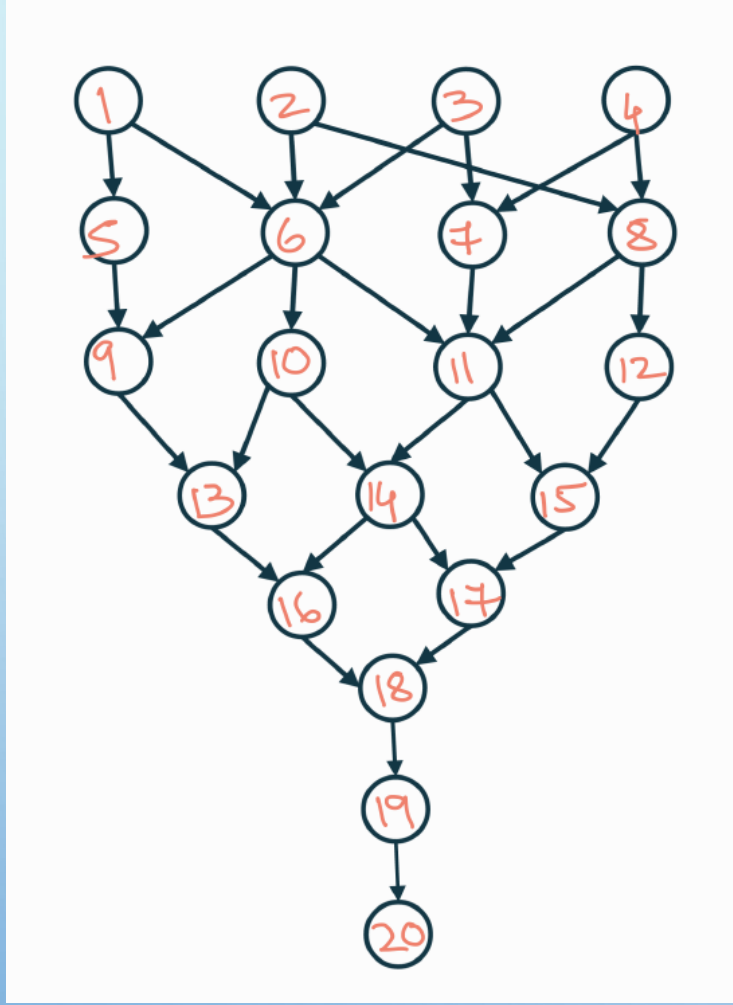
```
After Task Migration:
Task1: cloud, start time is: 0,finish time is: 5
Task2: cloud, start time is: 3,finish time is: 8
Task6: cloud, start time is: 6,finish time is: 11
Task3: cloud, start time is: 9,finish time is: 14
Task5: cloud, start time is: 12,finish time is: 17
Task4: local core1, start time is: 5,finish time is: 12
Task7: cloud, start time is: 15,finish time is: 20
Task10: cloud, start time is: 18,finish time is: 23
Task8: cloud, start time is: 21,finish time is: 26
Task14: cloud, start time is: 24,finish time is: 29
Task9: local core1, start time is: 17,finish time is: 22
Task11: local core2, start time is: 20,finish time is: 22
Task12: local core2, start time is: 26,finish time is: 29
Task13: local core1, start time is: 22,finish time is: 27
Task15: cloud, start time is: 29,finish time is: 34
Task19: cloud, start time is: 32,finish time is: 37
Task17: local core2, start time is: 29,finish time is: 34
Task18: local core3, start time is: 29,finish time is: 31
Task16: local core1, start time is: 29,finish time is: 35
Task20: cloud, start time is: 35,finish time is: 40
 Now the total energy is: 69
 Now the completion time is: 40
The running time of task migration of Graph 1 is 25.615 ms
```

OUTPUT OF THE THIRD EXAMPLE AFTER TASK MIGRATION



E1=7+5+5+6=23;
E2=(10)*2=20;
E3=2*4=8;
Es=(3*12*0.5)=18;
E=23+20+8+18=69

INPUT OF FOURTH EXAMPLE



Tmax = 40;

$$\begin{cases} T^S = 3; \\ T^C = 1; \\ T^R = 1; \end{cases}$$

$$\begin{cases} P_1 = 1; \\ P_2 = 2; \\ P_3 = 4; \\ P_s = 0.5; \end{cases}$$

Time in each core	local core1	local core2	local core3
Task 1	9	7	5
Task 2	8	6	5
Task 3	6	5	4
Task 4	7	5	3
Task 5	5	4	2
Task 6	7	6	4
Task 7	8	5	3
Task 8	6	4	2
Task 9	5	3	2
Task 10	7	4	2
Task 11	7	4	2
Task 12	6	3	2
Task 13	5	3	3
Task 14	8	7	4
Task 15	9	8	5
Task 16	6	4	2
Task 17	7	5	3
Task 18	8	4	2
Task 19	9	6	3
Task 20	7	6	4

INITIAL SCHEDULE

THE COMPLETION TIME OF THE SCHEDULE IS 33.
THE TOTAL ENERGY CONSUMED IS 179.5.
THE RUNNING TIME OF THE INITIAL SCHEDULE PROGRAM IS 0.067 ms.

```
Initial schedule:
Task1: local core3, start time is: 0,finish time is: 5
Task2: cloud, start time is: 0,finish time is: 5
Task3: local core2, start time is: 0,finish time is: 5
Task4: local core1, start time is: 0,finish time is: 7
Task6: local core3, start time is: 5,finish time is: 9
Task7: local core2, start time is: 7,finish time is: 12
Task8: local core3, start time is: 9,finish time is: 11
Task11: local core2, start time is: 12,finish time is: 14
Task12: local core3, start time is: 11,finish time is: 13
Task10: cloud, start time is: 9,finish time is: 14
Task5: local core1, start time is: 7,finish time is: 12
Task15: local core3, start time is: 14,finish time is: 19
Task14: cloud, start time is: 14,finish time is: 19
Task9: local core1, start time is: 12,finish time is: 17
Task13: local core2, start time is: 17,finish time is: 20
Task17: local core3, start time is: 19,finish time is: 22
Task16: local core2, start time is: 20,finish time is: 24
Task18: local core3, start time is: 24,finish time is: 26
Task19: local core3, start time is: 26,finish time is: 29
Task20: local core3, start time is: 29,finish time is: 33
 Now the total energy is: 179.5
 Now the completion time is: 33
The running time of initial schedule of Graph is 0.067 ms
```

OUTPUT OF THE FOURTH EXAMPLE AFTER INITIAL SCHEDULE

LOCAL CORE 1	4							5					9																		
LOCAL CORE 2	3							7					11					13				16									
LOCAL CORE 3	1					6			8				12			15				17				18		19			20		
WIRELESS SENDING	2								10						14																
CLOUD				2									10					14													
WIRELESS RECEIVING					2									10					14												
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

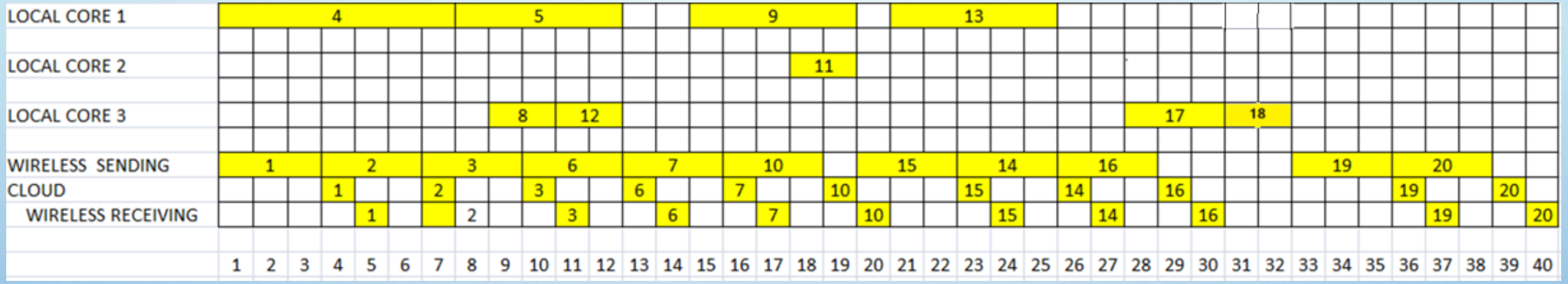
E1=7+5+5=17;
E2=(5+5+2+3+4)*2=38;
E3=30*4=120;
Es=(3*3*0.5)=4.5;
E=17+38+120+4.5=179.5

TASK MIGRATION

THE COMPLETION TIME OF THE SCHEDULE IS 40.
THE TOTAL ENERGY CONSUMED IS 78.5.
THE RUNNING TIME OF THE INITIAL SCHEDULE PROGRAM IS 32.09 ms.

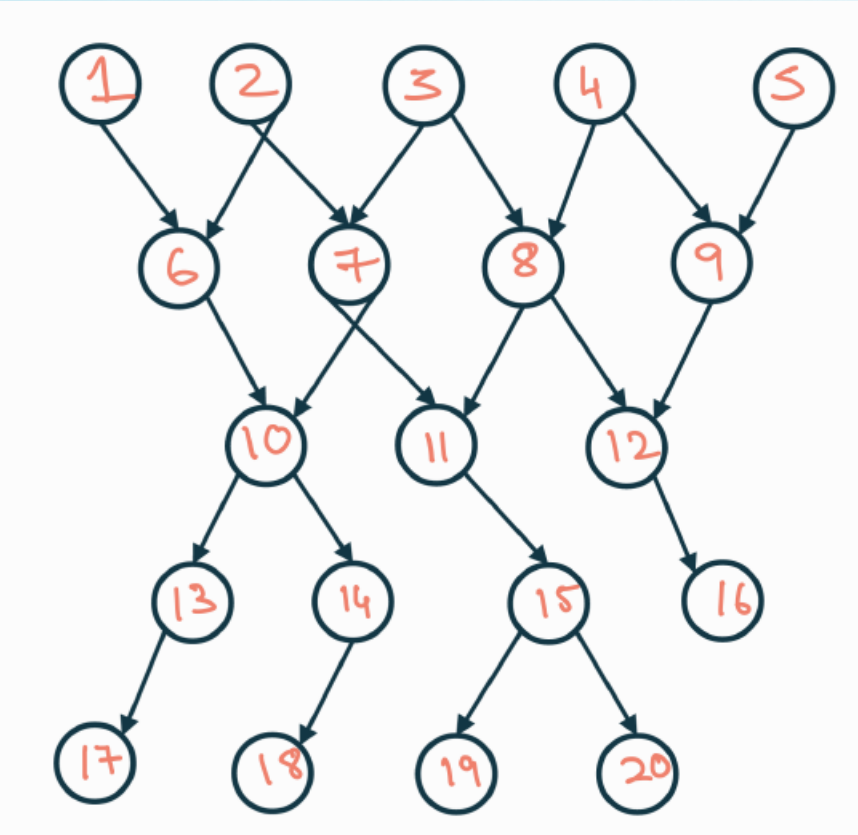
```
After Task Migration:
Task1: cloud, start time is: 0,finish time is: 5
Task2: cloud, start time is: 3,finish time is: 8
Task3: cloud, start time is: 6,finish time is: 11
Task4: local core1, start time is: 0,finish time is: 7
Task6: cloud, start time is: 9,finish time is: 14
Task7: cloud, start time is: 12,finish time is: 17
Task8: local core3, start time is: 8,finish time is: 10
Task11: local core2, start time is: 17,finish time is: 19
Task12: local core3, start time is: 10,finish time is: 12
Task10: cloud, start time is: 15,finish time is: 20
Task5: local core1, start time is: 7,finish time is: 12
Task15: cloud, start time is: 19,finish time is: 24
Task14: cloud, start time is: 22,finish time is: 27
Task9: local core1, start time is: 14,finish time is: 19
Task13: local core1, start time is: 20,finish time is: 25
Task17: local core3, start time is: 27,finish time is: 30
Task16: cloud, start time is: 25,finish time is: 30
Task18: local core3, start time is: 30,finish time is: 32
Task19: cloud, start time is: 32,finish time is: 37
Task20: cloud, start time is: 35,finish time is: 40
 Now the total energy is: 78.5
 Now the completion time is: 40
The running time of task migration of Graph 1 is 32.09 ms
```

OUTPUT OF THE FOURTH EXAMPLE AFTER TASK MIGRATION



E1=(7+5+5+5)*1=22;
E2=(2)*2=4;
E3=(2+2+3+2)*4=36;
Es=(3*11*0.5)=16.5;
E=22+4+36+16.5=78.5

INPUT OF Fifth EXAMPLE



Tmax = 40;

$$\begin{cases} T^S = 3; \\ T^C = 1; \\ T^R = 1; \end{cases}$$

$$\begin{cases} P_1 = 1; \\ P_2 = 2; \\ P_3 = 4; \\ P_s = 0.5; \end{cases}$$

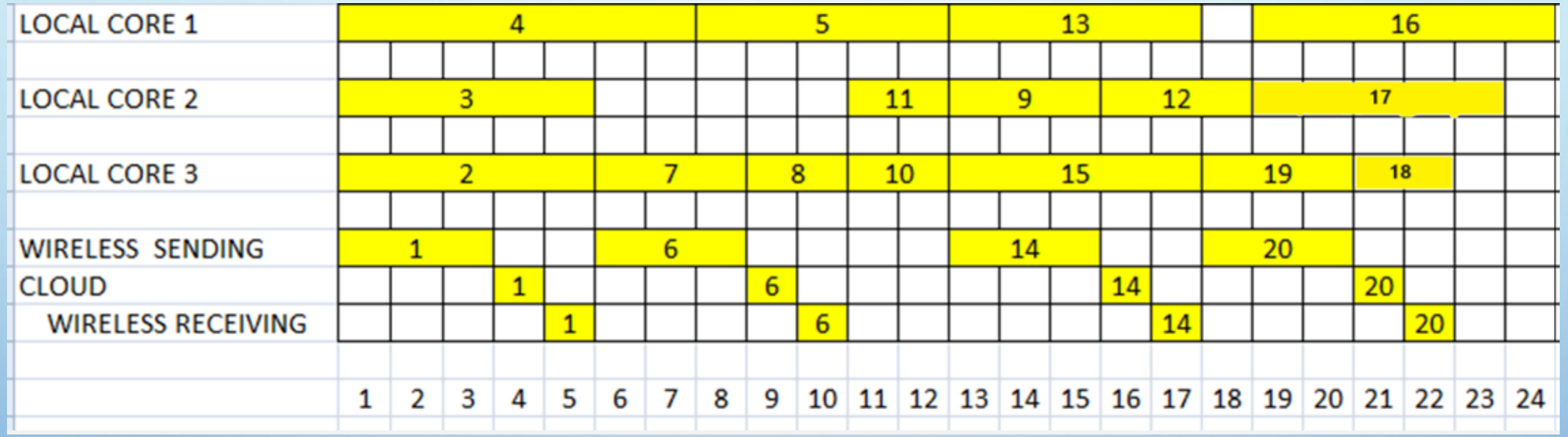
Time in each core	local core1	local core2	local core3
Task 1	9	7	5
Task 2	8	6	5
Task 3	6	5	4
Task 4	7	5	3
Task 5	5	4	2
Task 6	7	6	4
Task 7	8	5	3
Task 8	6	4	2
Task 9	5	3	2
Task 10	7	4	2
Task 11	7	4	2
Task 12	6	3	2
Task 13	5	3	3
Task 14	8	7	4
Task 15	9	8	5
Task 16	6	4	2
Task 17	7	5	3
Task 18	8	4	2
Task 19	9	6	3
Task 20	7	6	4

INITIAL SCHEDULE

The completion time of the schedule is 24.
The total energy consumed is 153.
The running time of the initial schedule program is 0.056 ms

```
Initial schedule:
Task2: local core3, start time is: 0,finish time is: 5
Task1: cloud, start time is: 0,finish time is: 5
Task3: local core2, start time is: 0,finish time is: 5
Task4: local core1, start time is: 0,finish time is: 7
Task7: local core3, start time is: 5,finish time is: 8
Task8: local core3, start time is: 8,finish time is: 10
Task6: cloud, start time is: 5,finish time is: 10
Task11: local core2, start time is: 10,finish time is: 12
Task10: local core3, start time is: 10,finish time is: 12
Task5: local core1, start time is: 7,finish time is: 12
Task15: local core3, start time is: 12,finish time is: 17
Task9: local core2, start time is: 12,finish time is: 15
Task14: cloud, start time is: 12,finish time is: 17
Task13: local core1, start time is: 12,finish time is: 17
Task12: local core2, start time is: 15,finish time is: 18
Task19: local core3, start time is: 17,finish time is: 20
Task20: cloud, start time is: 17,finish time is: 22
Task17: local core2, start time is: 18,finish time is: 23
Task18: local core3, start time is: 20,finish time is: 22
Task16: local core1, start time is: 18,finish time is: 24
 Now the total energy is: 153
 Now the completion time is: 24
The running time of initial schedule of Graph is 0.056 ms
```

OUTPUT OF THE FIFTH EXAMPLE AFTER INITIAL SCHEDULE



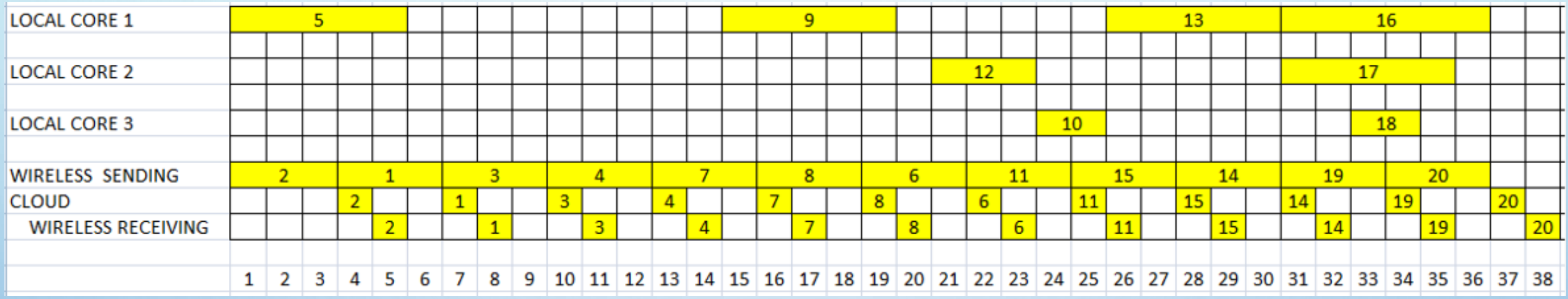
E1=(7+5+5+6)*1=23;
E2=(5+2+3+3+5)*2=36;
E3=(5+3+2+2+5+3+2)*4=88;
Es=(3*4*0.5)=6;
E=23+36+88+6=153

Task migration

The completion time of the schedule is 38.
The total energy consumed is 71.
The running time of the initial schedule program is 22.255 ms.

```
After Task Migration:
Task2: cloud, start time is: 0,finish time is: 5
Task1: cloud, start time is: 3,finish time is: 8
Task3: cloud, start time is: 6,finish time is: 11
Task4: cloud, start time is: 9,finish time is: 14
Task7: cloud, start time is: 12,finish time is: 17
Task8: cloud, start time is: 15,finish time is: 20
Task6: cloud, start time is: 18,finish time is: 23
Task11: cloud, start time is: 21,finish time is: 26
Task10: local core3, start time is: 23,finish time is: 25
Task5: local core1, start time is: 0,finish time is: 5
Task15: cloud, start time is: 24,finish time is: 29
Task9: local core1, start time is: 14,finish time is: 19
Task14: cloud, start time is: 27,finish time is: 32
Task13: local core1, start time is: 25,finish time is: 30
Task12: local core2, start time is: 20,finish time is: 23
Task19: cloud, start time is: 30,finish time is: 35
Task20: cloud, start time is: 33,finish time is: 38
Task17: local core2, start time is: 30,finish time is: 35
Task18: local core3, start time is: 32,finish time is: 34
Task16: local core1, start time is: 30,finish time is: 36
 Now the total energy is: 71
 Now the completion time is: 38
The running time of task migration of Graph 1 is 22.255 ms
```

OUTPUT OF THE FIFTH EXAMPLE AFTER TASK MIGRATION



E1=(5+5+5+6)*1=21;
E2=(3+5)*2=16;
E3=(2+2)*4=16;
Es=(3*12*0.5)=16.5;
E=21+16+16+18=71

PART 3 - CONCLUSION

- ACHIEVE THE GOAL WHICH IS TO MINIMIZE THE CONSUMPTION OF THE MOBILE DEVICE.
- APPLY ONE EXAMPLE IN PAPER [1].
- THE RESULT OF ANOTHER EXAMPLES ARE ABSOLUTELY CORRECT.
- THE RESULT OF THE FIRST EXAMPLE IS LITTLE DIFFERENT FROM PAPER [1]

REFERENCES

[1] X. LIN, Y. WANG, Q. XIE AND M. PEDRAM, "ENERGY AND PERFORMANCE-AWARE TASK SCHEDULING IN A MOBILE CLOUD COMPUTING ENVIRONMENT," *2014 IEEE 7TH INTERNATIONAL CONFERENCE ON CLOUD COMPUTING*, ANCHORAGE, AK, 2014, PP. 192-199, DOI: 10.1109/CLOUD.2014.35.