APPENDIX-C

1 Slot Scheduling

The scheduling problem is the following.

First partition the frame of size N slots into a contiguous part with N_D downlink slots and an uplink part with N_U uplink slots, such that $N_D + N_U = N - N_B$, where N_B is the number of slots required for the periodic beacon. Typically, $N_D \gg N_U$ as TCP data traffic is highly asymmetric since users download a lot more than they upload, and during downloads, long TCP packets (upto 1500 bytes) are received in the downlink and one 40 byte TCP ACK is sent in the uplink for alternate recieved packets.

Now, when $m_{v,i}, 1 \leq i \leq m$, VOIP calls are admitted for ST i, one needs to determine the number of slots C_i to be reserved in the uplink and downlink subframes for ST i, such that the QoS targets are met for all the voice calls. For doing this, evidently the set of vectors $\mathbf{C} = (C_1, \ldots, C_m)$ that are feasible (i.e., can be scheduled) needs to be known. For each deployment, there will be an optimal set of such vectors $\mathcal{C}_{opt}$, and for any practical scheduler, there will be an achievable set of admissible vectors $\mathcal{C} \in \mathcal{C}_{opt}$.

Once the required vector of voice payload slots has been scheduled, one has to schedule as many additional payload slots, so as to maximize the traffic carrying capacity for TCP while ensuring some fairness between the flows.

1.1 Representation of a Schedule

A feasible schedule for the deployment in Figure 1 is given in the table below, where each row stands for a sector and each colomn stands for a slots. There are 15 slots in the example. The entry in the table denotes the index of the ST that is transmitting in each sector in each slot. ST 1 in sector 2 and ST 2 in sector 1 transmit for the first 4 slots. At the end of 4th slot, ST 1 stops transmitting and ST 4 starts transmitting. The matrix representation of the schedule is also shown. The first 3 columns stands for the 3 sectors. Each entry shows the index of the ST that is transmitting. The last column indicates the number of slots for which that row is operative. Here, the sum of elements in column 4 is 15, indicating that there are 15 slots. This notation is followed throughout the examples.

slots $\rightarrow$

2	2	2	2	2	2	2	2	0	0	0	0	0	0	0
1	1	1	1	4	4	4	4	4	4	4	4	4	4	4
0	0	0	0	0	0	0	0	6	6	6	6	6	6	6

$$\begin{bmatrix} 2 & 1 & 0 & 4 \\ 2 & 4 & 0 & 4 \\ 0 & 4 & 6 & 7 \end{bmatrix}$$

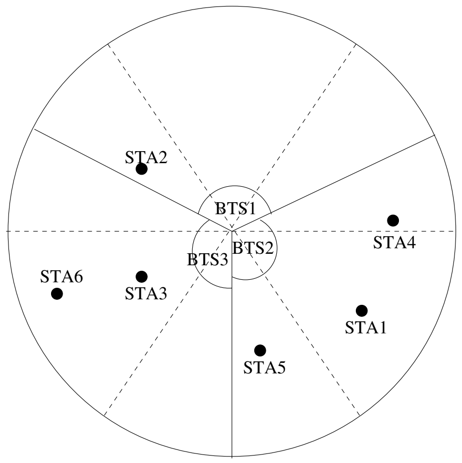

Figure 1: A system example showing the distribution of 6 stations in 3 sectors

1.2 A Greedy Heuristic Scheduler for the Uplink

The STs are scheduled such that the one with the longest queue is scheduled first. Find an activation vector which includes the ST with the largest voice queue. Next include a non interfering ST with the longest queue and so on until the number of STs in the activation set is equal to the number of simultaneous transmissions possible or till the activation set is maximal. A maximal activation set is one to which one cannot add any more links such that there is no interference between the links in the set. Use this maximal activation vector until one of the STs in the set completes transmission. Once one of the STs complete transmission, we remove that ST from the set and schedule another ST, that does not interfere with the STs in the set. Repeat the procedure until all the STs completes transmitting their voice packets. When all the STs complete their voice transmission, the remaining slots are used for TCP transmission. If at any stage during voice transmission, a situation occurs where there are no more noninterfering STs in a sector which can transmit voice, but there is one that can transmit data, schedule data for that interval.

In the beginning of each frame, for each slot k, heuristically build an activation vector $\mathbf{u}_k \in \mathcal{U}$ starting from an ST in $\{i : q_{k,i} = \max_j q_{k,j}\}$, i.e., the non interfering station with the maximum voice queue. Here, q_{kj} denotes the queue length of the jth ST at the beginning of the kth slot. k varies from 1 to N over a frame. Build a maximal activation vector beginning with that link, and augmenting the vector every time with a non interfering link.

$\mathcal{I}(\mathbf{u})$ denote the interference set of activation set $\mathbf{u}$, the set of links that can interfere with the STs in $\mathbf{u}$).

Algorithm 1.1

1. Modify the voice queue lengths to include the overhead slots required. i.e., If an ST has a voice queue of 2 packets, add 3 slots of PHY overhead to make the queue length 5.

2. Initially, slot index $k = 0$. Let ST i be such that

$$q_{ki} = \max_{l=1\ldots m} \{q_{kl}\}$$

i.e., The ST with longest voice queue at the beginning of slot k is i. Form activation vector $\mathbf{u}$ with link i activated. i.e., $\mathbf{u} = \{i\}$

3. Let ST j be such that
$$q_{kj} = \max_{l}\{q_{kl} : l \notin \mathcal{I}(\mathbf{u})\}$$
j is such that it is the non interfering ST with maximum queue length. Augment s with link j. Now, find $\mathcal{I}(\mathbf{u})$ corresponding to the new $\mathbf{u}$.

4. Repeat step 3 until activation vector that we get is a maximal activation vector.

5. Let
$$n = \{q_{kl} : \min_{l=1,\ldots,m}(q_{kl}, l \in \mathbf{u})\}$$
i.e., n is the minumum number of slots required for the first ST in $\mathbf{u}$ to complete its transmission. Use $\mathbf{u}$ in the schedule from kth to $(k+n)$th slot.
$$q_{k+n,i} = \begin{cases} q_{k,i} - n & \text{for} \quad i \in \mathbf{u}' \\ q_{k,i} & \text{for} \quad i \notin \mathbf{u}' \end{cases}$$
and $k = k+n$ i.e., slot index advances by n, and the queue length for the STs at the beginning of $k+n$th slot is n less

6. At the end of $k+n$th slot,
$$\mathbf{u} = \mathbf{u} - \{l : q_{kl} = min(q_{kl}, l \in \mathbf{u})\}$$
i.e., remove from the activation vector, those STs that have completed their voice slot requirement.

7. Go back to Step 3 and form maximal activation vector including $\mathbf{u}$. Continue the above procedure until $\mathbf{q} = \mathbf{0}$ or $n = N_U$ In this step, we form a new activation vector with the remaining STs in the activation vector (which need more slots to complete their requirement).

8. Once the voice packets are transmitted, we serve the TCP packets in the same way, except that if in forming a maximal activation set, it is found that the only schedulable ST has only TCP packets to send, then TCP packets are scheduled.

If $\mathbf{q} > \mathbf{0}$ when $n = N_U$, the allocation is infeasible.

1.3 A Greedy Heuristic Scheduler for the Downlink

The difference of the downlink scheduling problem from the uplink scheduling problem is that in downlink, a transport block can contain packets to multiple STs. By combining the voice packets to different STs to a single TB, we save considerable PHY overhead. For transmitting a single voice packet needs 4 slots, where 3 slots are for the PHY header. Transmitting 2 voice packets need only 5 slots. So, it is always advantageous to have transmissions in longer blocks. This can

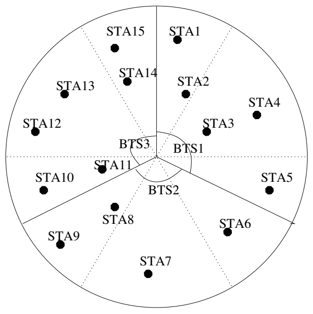

Figure 2: Typical deployment of a system with 3 sectors and 15 STs

be done by grouping together the STs to those which are heard only by ith BTS, those heard by ith and $(i-1)$th BTS, but associated to the ith BTS and those heard by ith and $(i-1)$th BTS, but associated to the $(i-1)$th BTS, for all values of i.

For example in Figure 2 STs 3 and 4 are associated with BTS 1 and hears only form BTS 1. So, any ST in the interference set of 3 will also be in the interference set of 4. Any transmission to ST 3 can equivalently be replaced by a transmission to 4. So, they form a group for the down likn schedule. Similarly, STs 8 and 9 are associated to BTS 2 and interfere with BTS 3. They are associated to the same BTS and cause interference to the same STs. So, ST 8 and 9 also form a group.

The STs are grouped together based on the above criterion. The queue length of each group would be the sum of queue lengths of the STs forming the group. The greedy heuristic scheduler for the uplink scheduling problem can then be used over these groups. This is made clearer in Example 2.3

1.4 Round Robin Scheduling

A scheduler with least complexity can be designed as follows. The uplink and downlink parts of the frame may further be divided into two contigous parts. Alternate sectors are served in these two parts. For example, Sectors 1, 3, 5 are served in the first part, and Sectors 2, 4, 6 can be served in the second part of the frame. Interference between adjacent sectors can be eliminated in this way. With the number of sectors close to $2n_0$, performance of this scheduler would be equivalent to that of the scheduler discussed in Section **??**, since we can have n_0 transmissions going on in each slot, with this scheduler. But, with $n_0 = 4$, this would require 8 sectors in the system. With number of sectors less than $2n_0$, the number of simultaneous transmissions would be less than n_0 with the round robin algorithm, where as we can have upto n_0 transmissions with the greedy algorithm.

1.5 Fair Scheduling

Having all voice transmission in the beginning and data afterwards is wasteful in terms of overhead. Following the principle of having longer TBs, it is advantageous to have the voice and data tranfer to an ST in a block. To provide fairness to users, keep track of the average rates allocated to STs over time. The STs with low average rate over frames are given a chance to transmit first. Maximal independent sets are formed starting from the ST with the lowest average rate. Once the slots for voice transmission are scheduled, the attempt should be to have TCP transmission in blocks of size T_{max}, so that PHY overhead per slot is minimised.

Let $\mathbf{R}_k$ be the vector of average rates allocated to STs till the kth frame and $\mathbf{r}_k$ be the vector of rates allocated to the STs in the kth slot, i.e, the fraction of slots allocated to STs in the kth frame. The avrage rate acheived by the STs is obtained by computing the following geometrically weighted average.A large value of α places lees weight on the previous frames.

$$\mathbf{R}_{k+1} = \alpha \mathbf{R}_k + (1-\alpha)\mathbf{r}_k$$

1. Given a rate vector $\mathbf{R}$, obtain a maximal independent set as follows
   (a) $\mathbf{u}_1 = \{i_1\}$
   $i_1 = \arg\min_{1 \le j \le n} \mathbf{R}_j$
   $\mathcal{I}(\mathbf{u}_1)$ is the set of links interfering with the links in $\mathbf{u}_1$. In this step, we select the ST with the smallest average rate R_k for transmission.
   (b) Choose $i_2 \in \arg\min_{1 \le j \le n, i_2 \notin \mathcal{I}(\mathbf{u_1})} \mathbf{R}_j$
   $\mathbf{u}_1 = \{i_1, i_2\}$. In this step, we select one of the non interfering STs with minimum average rate for transmission.
   (c) Repeat the above until a maximal independent set is obtained. Now, we have a set with STs which have received low average rates in the previous slots. So, once all STs transmit their voice packets, we schedule these STs for data packets.
2. Let l_1 denote the number of nodes in $\mathbf{u_1}$ at the end of step 1. Repeat the above for the remaining $n - l_1$ nodes. Now we have a maximal independent set from the remaining $N_u - l_1$ nodes. If any one of the l_1 nodes can be activated along with the maximal independent set formed from the $N_u - l_1$ nodes, add that till one get a maximal independent set. This yields $\mathbf{u}_1, \mathbf{u}_2 \ldots \mathbf{u}_k$ such that each node is included atleast once. Each node is included atleast once since a given number of slots is to be reserved for each ST in every frame.
3. Now, we need to schedule $\mathbf{u}_1$ for t_1, $\mathbf{u}_2$ for t_2, etc. To maximize throughput, we take $t_j = T_{max}$ or number of voice slots required. The vectors in the initial part of the schedule had low average rate over frames. So, they get priority to send data packets. So, starting from j=1, i.e., from the first activation vector, if the sum of number of slots allocated to STs in the frame is less than N_u, $t_j = T_{max}$. Else, $t_j =$ number of voice slots required.Therefore transmission takes place in blocks of length equal to T_{max} as long as it is possible.
4. Update the rate vector as
$$\mathbf{R}_{k+1} = \alpha \mathbf{R}_k + (1-\alpha)\mathbf{r}_k$$

2 Design Methodology: Examples

The deployment considered in the examples are described by the $\mathbf{A}$ and $\mathbf{I}$ matrices given in Table **??**. The $\mathbf{A}$ matrix shows the BTS to which each ST is associated. Columns in both matrices correspond to sectors and rows correspond to STS. $\mathbf{A}_{ij} = 1$ indicates that the ith ST is associated to the jth BTS. Each ST is associeted to one and only one BTS. $\mathbf{I}_{ij} = 1$ indicates that the ith ST can hear from the jth BTS. An ST might hear from more than one BTS.

Example 2.1

Equal Voice Load

In this example there are 40 stations in 5 sectors. Any station within 72° is associated with the BTS and any station within 32° of the center of a mainlobe can cause interference at that BTS. We take the maximum burst length, T_{max} to be 15 and $N_U = 112$. Voice packet (including MAC header) is 44 bytes long. So, each voice packet requires one slot time for transmission. In this example, we take $n_0 = 4$.

Consider all stations to have the same voice slot requirement of 4 packets. The uplink transmission from each station could be done in a single block , which requires a total of 3 slots of overhead. So, the schedule should give at least 7 slots for each station. One schedule that attains this, (as obtained Algorithm **??**) is given by $\mathbf{S}_2$.

The first 5 columns of the matrix $\mathbf{S}_2$ gives the links that can transmit in each sector. A 0 indicates that there cannot be a transmission by any of the stations in this sector without causing interference to at least one of the ongoing transmissions. The entries in the 6th column indicates the number of consecutive slots for which these vector is used in the schedule. The 7th column is the number of slots (over all the sectors) used for transmitting voice packets. The last column shows the number of slots, over all the sectors used for transmitting TCP packets.

For example, consider the first row of $\mathbf{S}_2$. This indicates that the ST 1 from Sector 1, 2 from Sector 2, 11 from Sector 3, 6 from Sector 4 and none from Sector 5 are scheduled for the first 7 slots. At the end of 7 slots, the voice queues of these stations are exhausted, so we schedule other stations.

From $\mathbf{S}_2$, one sees that ST 40 gets a chance for transmission only in the last row, i.e., by the 98th slot. It transmits till the 105th slot to complete transmitting its voice queue. This indicates that the scheduling of voice queues requires 105 slots.

In the 1st row, ST 11 has been scheduled for 7 slots. So the voice queue of ST 11 is exhausted by the end of that row. But, since there are no other STs in Sector 3, that can use the slot for voice transmission, we schedule ST 11 for TCP transmission further in the schedule. In this algorithm, priority is to STs that had been transmitting in the previous slot, and next priority to stations with lower index. But, better fairness can be ensured by round robining between the STs in a sector.

Since the voice queue of ST 11 is exhausted by the end of 1st row in $\mathcal{S}_2$ the ST 11 scheduled after 1st row, i.e., from 29th slot is for TCP transmission. Similarly, ST 7 scheduled after the 11th row, ST 39 scheduled after 9th row etc. are for TCP transmission.

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0; \end{bmatrix} \quad \mathbf{Q}_1 = \begin{bmatrix} 3 \\ 2 \\ 3 \\ 4 \\ 5 \\ 5 \\ 2 \\ 4 \\ 4 \\ 5 \\ 3 \\ 2 \\ 5 \\ 2 \\ 4 \\ 5 \\ 5 \\ 4 \\ 4 \\ 5 \\ 3 \\ 2 \\ 4 \\ 4 \\ 3 \\ 5 \\ 5 \\ 3 \\ 4 \\ 5 \\ 2 \\ 5 \\ 4 \\ 5 \\ 5 \\ 3 \\ 3 \\ 2 \\ 5 \\ 4 \end{bmatrix} \quad \mathcal{S}_1 = \begin{bmatrix} 0 & 10 & 15 & 6 & 5 & 7 & 28 & 0 \\ 0 & 10 & 24 & 6 & 5 & 1 & 4 & 0 \\ 16 & 0 & 24 & 13 & 27 & 6 & 24 & 0 \\ 16 & 39 & 0 & 13 & 27 & 2 & 8 & 0 \\ 32 & 39 & 11 & 20 & 0 & 6 & 24 & 0 \\ 32 & 4 & 12 & 20 & 0 & 2 & 8 & 0 \\ 8 & 4 & 12 & 0 & 26 & 3 & 12 & 0 \\ 8 & 4 & 11 & 0 & 26 & 2 & 6 & 2 \\ 8 & 19 & 11 & 0 & 26 & 2 & 6 & 2 \\ 28 & 19 & 11 & 0 & 26 & 1 & 3 & 1 \\ 28 & 19 & 0 & 34 & 30 & 4 & 16 & 0 \\ 28 & 18 & 0 & 34 & 30 & 1 & 4 & 0 \\ 7 & 18 & 0 & 34 & 30 & 3 & 12 & 0 \\ 7 & 18 & 0 & 35 & 25 & 2 & 8 & 0 \\ 31 & 18 & 0 & 35 & 25 & 1 & 4 & 0 \\ 31 & 0 & 40 & 35 & 25 & 3 & 12 & 0 \\ 31 & 0 & 40 & 35 & 38 & 1 & 4 & 0 \\ 7 & 0 & 40 & 35 & 38 & 1 & 3 & 1 \\ 7 & 0 & 40 & 36 & 38 & 2 & 6 & 2 \\ 17 & 0 & 11 & 36 & 38 & 1 & 3 & 1 \\ 17 & 0 & 11 & 36 & 38 & 3 & 6 & 6 \\ 17 & 0 & 11 & 37 & 38 & 4 & 8 & 8 \\ 9 & 21 & 11 & 37 & 0 & 2 & 6 & 2 \\ 9 & 21 & 11 & 37 & 0 & 4 & 8 & 8 \\ 9 & 2 & 11 & 37 & 0 & 1 & 2 & 2 \\ 0 & 2 & 11 & 37 & 23 & 4 & 8 & 8 \\ 0 & 2 & 11 & 37 & 23 & 3 & 21 & 7 \\ 7 & 2 & 11 & 0 & 29 & 7 & 21 & 7 \\ 7 & 2 & 11 & 33 & 0 & 7 & 21 & 7 \\ 1 & 2 & 11 & 33 & 0 & 6 & 18 & 6 \\ 1 & 2 & 11 & 3 & 0 & 6 & 18 & 6 \\ 1 & 2 & 14 & 0 & 0 & 5 & 15 & 5 \\ 1 & 2 & 0 & 22 & 0 & 5 & 15 & 5 \end{bmatrix}$$

$$\mathbf{Q}_2 = \begin{bmatrix} 4 \\ 4 \\ 4 \\ \vdots \\ 4 \end{bmatrix} \quad \mathcal{S}_2 = \begin{bmatrix} 1 & 2 & 11 & 6 & 0 & 7 & 28 & 0 \\ 7 & 4 & 12 & 3 & 0 & 7 & 28 & 0 \\ 8 & 0 & 15 & 13 & 5 & 7 & 28 & 0 \\ 9 & 19 & 24 & 34 & 0 & 7 & 28 & 0 \\ 0 & 10 & 11 & 35 & 25 & 7 & 28 & 0 \\ 16 & 21 & 14 & 0 & 27 & 7 & 28 & 0 \\ 17 & 0 & 11 & 36 & 30 & 7 & 21 & 7 \\ 28 & 18 & 0 & 37 & 38 & 7 & 28 & 0 \\ 31 & 39 & 11 & 20 & 0 & 7 & 21 & 7 \\ 32 & 39 & 0 & 22 & 5 & 7 & 21 & 7 \\ 0 & 39 & 0 & 22 & 23 & 7 & 7 & 14 \\ 7 & 39 & 11 & 0 & 26 & 7 & 7 & 21 \\ 7 & 39 & 11 & 0 & 29 & 7 & 21 & 7 \\ 7 & 39 & 11 & 33 & 0 & 7 & 21 & 7 \\ 7 & 0 & 40 & 33 & 0 & 7 & 14 & 7 \end{bmatrix}$$

Table 1: Data and results for the Example 2.1 and 2.2. $\mathcal{S}_i$ is the schedule for voice slot requirements given by $\mathbf{Q}_i, i \in \{1, 2\}$. Note that the last two columns of $\mathcal{S}_i$ are the total number of slots in all sectors for voice and TCP packets respectively scheduled in each row.

In this example, 4 STs transmit in each slot (i.e., 3×26 slots of transmission.). Thereby, of the 112×5 slots available (since in each slot, at most one ST from a sector can transmit.),112×4 slots are used for transmission, giving a throughput of 78% and a goodput efficiency of 46%. In this schedule, transmission of TCP packets occur in 105 slots.

Example 2.2

Unequal Voice Load

Also given is the schedule obtained by the Algorithm **??**, when the voice slot requirements are different for different stations. Given is one such vector ($\mathbf{Q}_1$) and the corresponding schedule ($\mathbf{S}_1$).

In the example, in the first row, i.e., links 10,15,6,5 are scheduled for the first 7 slots. At the end of the 7th slot, ST 15 exhausts its voice queue. So, remove ST 15 from the vector, and add ST 24 which can take its place. This continues till one of the other queues become empty. Then, remove the one whose queue is empty, and schedule some other ST which does not interfere with the ongoing transmissions, and so on till all STs are served.

All the STs get a chance to transmit by the last row of the given schedule. But, the last row finishes transmission by the 108th slot (as seen by summing up the last column of $\mathbf{S}_1$). The slots from 108 to 112 may be used for transmitting TCP packets for the STs in the last row. Here, the throughput 78% and a goodput efficiency of 46%. In this schedule, transmission of TCP packets occurs in 147 slots.

Example 2.3

Downlink
In downlink, transmission to multiple STs can be done in a block. In the deployment in this example, STs 2, 4, 19, 21, 39 all hear only from BTS 2. This can be seen from the $\mathbf{A}$ and $\mathbf{I}$ matrices. All these STs have 1 in the second column, indicating that they are associated to BTS 2. The $\mathbf{I}$ matrix has 1s only in the seccond column, indicationg that they hear only from BTS 2. These are grouped in to b. The transmissions to these STs can take place in a TB, since an ST which interferes with any ST in h will interfere with every other ST in h, and vice versa. So every ST in h is equivalent with respect to interference constraints. An h occuring in the schedule in Table 2 indicates a transmission to this group of STs. The STs which are in the same group are associated to the same BTS and are in the exclusion region of same sectors. The groups so formed for the example are given in Table 2. Similarly, the STs 3,20 and 33 hear from BTSs 4 and 5; all of them are associated to BTS 4. They are grouped to c. All STs in c are equivalent with respect to the interference constraints. The transmissions to these STs can occur in a TB. Now, the voice slot requirement of a group is the sum of the voice slot requirements of individual STs constituting the group. The same algorithm as for the uplink is then used over the groups $a, b, c, d, \dots$ with their respective queue lengths to obtain the schedule in downlink. Aggregating STs in the association and exclusion regions of each BTS like this has the advantage of increasing the overall system throughput, since transmissions occur in longer TBs. The goodput efficiency in downlink is about 64%.

$$\mathbf{Q}_{2d} = \begin{bmatrix} 2 \\ 2 \\ \vdots \\ 2 \end{bmatrix} \quad \mathcal{S}_{2d} = \begin{bmatrix} f & b & 0 & e & d & 31 \\ f & b & h & e & 0 & 5 \\ f & b & h & c & 0 & 18 \\ a & b & h & c & 0 & 13 \\ f & b & h & 0 & n & 13 \\ 0 & g & h & 0 & n & 8 \\ 0 & g & i & 0 & n & 8 \\ j & 0 & i & 0 & n & 8 \\ f & k & 0 & 0 & n & 8 \\ f & k & 0 & l & d & 8 \\ 0 & k & 0 & l & m & 8 \\ 0 & 0 & o & e & m & 8 \end{bmatrix} \quad \begin{array}{lll} a = \{1, 9\} & b = \{2, 4, 19, 21, 39\} & c = \{3, 20, 33\} \\ d = \{5, 25, 27, 30, 38\} & e = \{6, 13, 34, 35, 36, 37\} & f = \{7, 8, 16, 28.31.32\} \\ g = \{10\} & h = \{11, 12, 15, 24\} & i = \{14\} \\ j = \{17\} & k = \{18\} & l = \{22\} \\ m = \{23\} & n = \{26, 29\} & o = \{40\} \end{array}$$

Table 2: The part of the downlink schedule for completing the voice slot requirement when 2 slots are reserved for voice transmission from each ST.