Throughout-Optimal Scheduling

What is the right way to select the actions $\vec{p}(t)$ to be taken across the links in our wireless network model? The answer to this question depends on the objective of the system. We will here present a scheduling scheme that obtains *throughput optimality*. In doing so, we will present some of the key engineering ideas in network scheduling. We will also discuss implications of our scheduling scheme to a few real-world network architectures.

Scheduling Policy

In any scheduling scheme, we need to introduce the concept of **queuing**. In our context, a queue is a "waiting area" where received transmissions are stored before they are forwarded over a link.

Formally, let q_l denote the queue length of link l. We are considering a slotted system here, where time is indexed by t, and without loss of generality, we assume the length of a timeslot is 1:

- The channel state $K(t)$ is fixed within a timeslot.
- Our scheme uses a fixed action $\vec{p}(t)$ within a timeslot.
- The capacity of each link is also fixed within a timeslot:

$$\vec{r}(t) = g(\vec{p}(t), K(t)) \tag{1}$$

The queue length then evolves as

$$q_l(t+1) = \Big[q_l(t) + \sum_s H_s^l A_s(t) - r_l(t)\Big]^+, \tag{2}$$

where $A_s(t)$ is the random arrivals to link l at time t, with $\mathbb{E}[A_s(t)] = X_s$, and $[x]^+ = \max\{0, x\}$. $r_l(t)$ can be thought of as the "drain rate" of the queue. We let $\vec{q}(t) = (q_1(t), ..., q_L(t))$ denote the vector of queue lengths.

The key intuition is that we need to make sure the queues do not explode to infinity. To drain queues faster, we would like to choose $\vec{p}(t)$ such that $\vec{r}(t)$ is large. If the current rate of link l cannot support the offered load, q_l will increase, so we should increase r_l, possibly at the cost of other links.

Consider the following policy: Pick $\vec{p}(t)$ at each time t such that

$$\vec{p}(t) = \underset{\substack{\vec{p}\in\mathcal{H} \\ \vec{r}=g(\vec{p},K(t))}}{\arg\max} \sum_l q_l(t) \cdot r_l. \tag{3}$$

This gives larger weight to r_l if $q_l(t)$ is large, i.e., to prioritize this link more at time t. We will show that this scheduling policy will be able to stabilize all queues for any offered load $\vec{X} \in \Lambda$.

It is important to note that this policy does not require knowledge of the flow rates $\vec{X}$ or the channel state distributions λ_k. It only needs the current channel state $K(t)$:

- The policy is queue-length based, which provides current state information about the system.
- The policy is adaptive based on the current network conditions. Further, this is an online solution, as decisions are made in each timeslot.

Before proving optimality, we can briefly consider why some other policies would definitely *not* work:

1. Since our goal is throughput maximization, suppose we choose the schedule that maximizes $\sum_l r_l$, or $\sum_l w_l r_l$ for fixed w_l. This would end up prioritizing the links with the highest achievable rates, with queues building up at other links.
2. On the other extreme, suppose we focus entirely on the queue length, and choose the schedule that maximizes $\sum_l q_l(t)\mathbb{1}_{\{p_l(t)>0\}}$. In other word, the objective function improves when we schedule links that have larger queues. But merely scheduling transmissions on a link says nothing about the rate achieved on that link.

Therefore, it is important to consider both the rates $\vec{r}(t)$ and the queues $\vec{q}(t)$ in making scheduling decisions.

Lyapunov Stability

Our main theoretical approach will draw from **Lyapunov stability**. Roughly speaking, a system is said to be Lyapunov stable if a solution starting near an equilibrium point will stay near the equilibrium forever.

Formally, we will seek a Lyapunov function $V(\vec{q})$ of the queue $\vec{q}$ with the following properties:

1. $V(\vec{q}) \geq 0$ for all $\vec{q}$, and $V(\vec{q}) \to +\infty$ as $||\vec{q}|| \to +\infty$.
2. Whenever $||\vec{q}(t)|| \geq M$ for some $M > 0$,
$$\mathbb{E}\big[V(\vec{q}(t+1)) - V(\vec{q}(t)) \mid \vec{q}(t)\big] \leq -\epsilon||\vec{q}(t)|| \tag{4}$$
for some $\epsilon > 0$.

The first property is pretty straightforward to satisfy, and establishes that the Lyapunov function $V(\vec{q})$ will grow with the size of $\vec{q}$. The second property then states that whenever $V(\vec{q})$ grows large, it will experience a **negative drift** proportional to the the size of $\vec{q}$. This implies that $\vec{q}(t)$ cannot explode to infinity. Visually, if we consider contour plots of $V(\vec{q})$, once we arrive at a point $V(\vec{q}) > M$, the system will experience a negative drift towards back inside the region:

Note also that the negative drift condition can be relaxed to

$$\mathbb{E}\big[V(\vec{q}(t+1)) - V(\vec{q}(t)) \mid \vec{q}(t)\big] \leq -\epsilon||\vec{q}(t)|| + M, \tag{5}$$

where M and ϵ can be taken as any positive constants. Their exact values are not important for our purpose here, as we only need to ensure that the drift becomes negative once $||\vec{q}||$ becomes large, and that it is proportional to $||\vec{q}||$. Thus, when $M \leq \frac{\epsilon}{2}||\vec{q}(t)||$, we can express the condition as

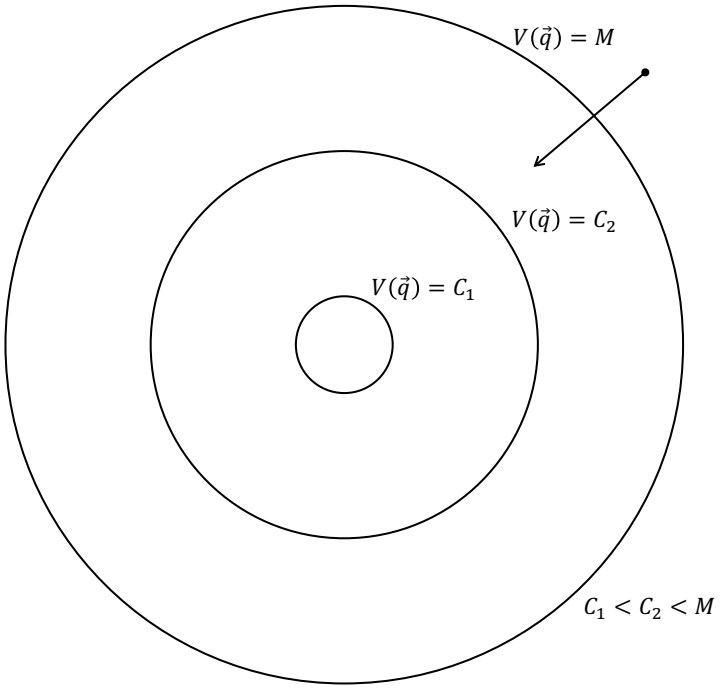
$$\mathbb{E}\big[V(\vec{q}(t+1)) - V(\vec{q}(t)) \mid \vec{q}(t)\big] \leq -\frac{\epsilon}{2}||\vec{q}(t)||. \tag{6}$$

When we derive the drift, we can ignore all terms that are bounded, and focus on those terms that grow with $\vec{q}$.

Proof of Throughput-Optimality

We will show that the function

$$V(\vec{q}) = \frac{1}{2}\sum_{l} q_l^2 \tag{7}$$



can serve as the Lyapunov function, and that the policy in (3) corresponds to maximizing the negative drift of this function.

Proof: Based on the queue evolution in (2), we have

$$\begin{aligned}(q_l(t+1))^2 &\leq \big(q_l(t) + \sum_s H_s^l A_s(t) - r_l(t)\big)^2 \quad (8)\\ &= (q_l(t))^2 + 2q_l(t)\Big[\sum_s H_s^l A_s(t) - r_l(t)\Big] \\ &\quad + \Big[\sum_s H_s^l A_s(t) - r_l(t)\Big]^2 \quad (9)\end{aligned}$$

Since $A_s(t)$ is finite and $r_l(t)$ is bounded in practice, the rightmost term will always be finite, and there exists a constant M_l such that

$$(q_l(t+1))^2 \leq (q_l(t))^2 + 2q_l(t)\Big[\sum_s H_s^l A_s(t) - r_l(t)\Big] + M_l^2. \quad (10)$$

Based on the definition of $V(\vec{q})$ in (7), then,

$$V(\vec{q}(t+1)) \leq V(\vec{q}(t)) + \sum_l q_l(t)\Big[\sum_s H_s^l A_s(t) - r_l(t)\Big] + \frac{1}{2}\sum_l M_l^2. \quad (11)$$

Applying the expectation from the left side of (6),

$$\mathbb{E}\big[V(\vec{q}(t+1)) - V(\vec{q}(t)) \mid \vec{q}(t)\big] \;\leq\; \sum_l q_l(t)\Big[\sum_s H_s^l X_s\Big] - \underbrace{\sum_l q_l(t)\mathbb{E}\Big[r_l(t) \mid q_l(t)\Big]}_{(a)} + \frac{1}{2}\sum_l M_l^2. \tag{12}$$

We see that term (a) matches the objective function in our policy (3). In other words, the max-weight policy is chosen to minimize the negative drift in the Lyapunov function $V(\vec{q})$.

Now, we must show that the resulting drift satisfies (6). Recall the expression for the capacity region from the last lecture:

$$\Lambda = \sum_k \lambda_k \cdot \mathsf{Conv\_hull}\{g(\vec{p},k) \mid \vec{p} \in \mathcal{H}\}. \tag{13}$$

Consider any $\vec{X}$ that lies strictly inside Λ. Since the region is convex, then a small perturbation $(1+\epsilon)\vec{X}$ for some $\epsilon > 0$ will lie in the region as well. Since this point is in Λ, there must exist a collection of actions $\vec{p}_k^{\,m} \in \mathcal{H}$ with fractions α_k^m and $\sum_m \alpha_k^m = 1$ such that:

$$(1+\epsilon)\Big[\sum_s H_s^l X_s\Big] \leq \sum_k \lambda_k \Big(\sum_m \alpha_k^m g(\vec{p}_k^{\,m},k)\Big) \qquad \text{for all } l. \tag{14}$$

Ultimately, we need to bound $(1+\epsilon)\Big[\sum_s H_s^l X_s\Big]$ in terms of the negative drift term (a) in (12). To do this, we will look for an upper bound on the right hand side of (14). With our policy (3) in place, at any time t,

$$\sum_l q_l(t) r_l(t) = \max_{\vec{r}=g(\vec{p},K(t))} \sum_l q_l(t)\cdot r_l. \tag{15}$$

Thus, for any action m and any channel state k,

$$\sum_l q_l(t) g_l(\vec{p}_k^{\,m},k)\mathbb{1}_{\{K(t)=k\}} \leq \sum_l q_l(t) r_l(t)\mathbb{1}_{\{K(t)=k\}}, \tag{16}$$

where $g_l(\vec{p}_k^{\,m},k)$ is the rate obtained on link l from $\vec{p}_k^{\,m}$, and $r_l(t)$ is the rate from the optimal action. Taking the expected value of both sides of this expression conditioned on $\vec{q}(t)$, we have

$$\sum_l q_l(t)\Big[\sum_k \lambda_k\big(\sum_m \alpha_k^m g_l(\vec{p}_k^{\,m},k)\big)\Big] \;\leq\; \sum_l q_l(t)\mathbb{E}[r_l(t)\mathbb{1}_{\{K(t)=k\}} \mid \vec{q}(t)]$$

$$= \sum_l q_l(t)\mathbb{E}[r_l(t) \mid \vec{q}(t)], \qquad (17)$$

where the indicator functions become absorbed by the expectation over channel states. Now, we can combine this with (14) by summing both sides of (14) over links to fit the right hand side of (14) with the left hand side of (17):

$$(1+\epsilon)\sum_l q_l(t)\Big[\sum_s H_s^l X_s\Big] \leq \sum_l q_l(t)\mathbb{E}[r_l(t) \mid \vec{q}(t)] \qquad (18)$$

Finally, using (18) back in (12), we have

$$\begin{aligned}
\mathbb{E}\big[V(\vec{q}(t+1)) - V(\vec{q}(t)) \mid \vec{q}(t)\big] &\leq -\epsilon\sum_l q_l(t)\Big[\sum_s H_s^l X_s\Big] + \frac{1}{2}\sum_l M_l^2 \\
&\leq -\epsilon\Big[\min_l \sum_s H_s^l X_s\Big]\Big(\sum_l q_l(t)\Big) + \frac{1}{2}\sum_l M_l^2 \\
&\leq -\frac{\epsilon}{2}\Big[\min_l \sum_s H_s^l X_s\Big]\Big(\sum_l q_l(t)\Big), \qquad (19)
\end{aligned}$$

where this last step comes from assuming $\sum_l M_l^2 \leq \epsilon(\min_l \sum_s H_s^l X_s)\sum_l q_l(t)$, or in other words,

$$\sum_l q_l(t) \geq \frac{\sum_l M_l^2}{\epsilon\Big[\min_l \sum_s H_s^l X_s\Big]}. \qquad (20)$$

Based on (19) and (20), the Lyapunov condition (6) holds as soon as the size of $\vec{q}(t)$ rises above a certain threshold. Thus, our policy in (3) stabilizes queues for any offered load in Λ, giving throughput optimality.

We next will walk through the implications of our scheduling scheme in two types of networks.

Cellular Networks

Recall previously we argued that for data traffic in a single cell, FDMA/TDMA schemes obtain better performance than CDMA. We will show now that our scheduling scheme (3) is consistent with this argument.

(1) FDMA/TDMA/OFDM: Consider a single cell, and first assume that the BS can schedule one user on any time/frequency channel, i.e., FDMA, TDMA, or OFDM.

The throughput-optimal scheme here reduces to picking the user with the largest $q_l(t)r_l(t)$ at time t, where $r_l(t)$ is the rate available to user l if they are scheduled. We assume here that interference from other cells is fixed.

If there are multiple channels/sub-carriers available for each user, then on each channel/sub-carrier c, we schedule the user l with the largest

$$q_l(t)r_l^c(t), \tag{21}$$

where $r_l^c(t)$ is the rate available to user l if it is scheduled in channel c. Note that $r_l^c(t)$ could vary over c due to frequency-selective fading.

- Suitable for TDMA/FDMA or OFDM systems.
- What if the BS can adjust power?

(2) CDMA: On the other hand, what if the BS uses some sort of CDMA scheme? In this case, more than one user can be scheduled at the same time.

We will focus our analysis here on the downlink. Assume that the rate is a linear function of the SINR, i.e.,

$$r_l(t) = B \cdot \frac{p_l g_l}{\sum_{k \neq l} p_k g_l + n_l}, \tag{22}$$

where p_l is the transmitting power (i.e., action) for user l, and g_l is the propagation gain to user l. We also assume that the BS has a total power constraint:

$$\sum_l p_l \leq p_{\max}. \tag{23}$$

From (22), the throughput-optimal policy then corresponds to

$$\underset{\vec{p}:\sum_{l=1}^{L} p_l \leq p_{\max}}{\arg\max} \quad f(\vec{p}) = \sum_{l=1}^{L} q_l \cdot \frac{p_l g_l}{\sum_{k \neq l} p_k g_l + n_0}. \tag{24}$$

It turns out that the solution to this optimization problem is of the following form: the BS will transmit to one user at the maximum power.

To see this, suppose there exists two users l and k and the BS is transmitting to them at the same time, i.e., $p_l > 0$, $p_k > 0$. Let $p_0 = p_l + p_k$, and let $p_l = x$, $p_k = p_0 - x$. We consider the behavior of $f(\vec{p})$ as a function of x:

$$f(x) = f(\ldots, \underbrace{x}_{p_l}, \ldots, \underbrace{p_0 - x}_{p_k}, \ldots) \tag{25}$$

It turns out that $f(x)$ is a convex function of $p_l = x$. Specifically, note that $f(x) = f(p_l)$ has three kinds of summation terms $h = 1, ..., L$:

(a) $h \neq l, h \neq k$. These terms are expressed as

$$q_h \cdot \frac{p_h g_h}{\sum_{m \neq h} p_m g_h + n_0} = q_h \cdot \frac{p_h g_h}{\sum_{m \neq h,l,k} p_m g_h + p_0 g_h + n_0}, \tag{26}$$

which is independent of x (since $p_l + p_k = p_0$). Thus, these terms are constant with respect to x.

(b) $h = l$. This term is expressed as

$$\begin{aligned} q_l \cdot \frac{p_l g_l}{\sum_{m \neq l} p_m g_l + n_0} &= q_l \cdot \frac{x g_l}{\sum_{m \neq l,k} p_m g_l + (p_0 - x) g_l + n_0} \\ &= q_l \left[-1 + \frac{\sum_{m \neq l,k} p_m g_l + p_0 g_l + n_0}{\sum_{m \neq l,k} p_m g_l + (p_0 - x) g_l + n_0} \right], \end{aligned} \tag{27}$$

where this last step follows from the identity that

$$\frac{ax}{b - ax} = -1 + \frac{b}{b - ax}, \tag{28}$$

with $a = g_l$ and $b = \sum_{m \neq l,k} p_m g_l + p_0 g_l + n_0$. Now, the function in (28) is convex as long as $x \leq b/a$, which can be seen in the visual plot below). This condition on x is always true in our case since $p_l \leq \sum_m p_m$. Thus, term (27) is convex with respect to x.

(c) $h = k$. This term is expressed as

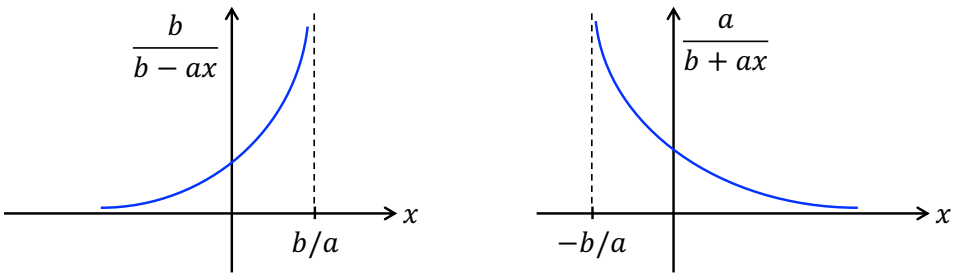
$$\begin{aligned} q_k \cdot \frac{p_k g_k}{\sum_{m \neq k} p_m g_k + n_0} &= q_k \cdot \frac{(p_0 - x) g_k}{\sum_{m \neq l,k} p_m g_k + x g_k + n_0} \\ &= q_k \left[-1 + \frac{\sum_{m \neq l,k} p_m g_k + p_0 g_k + n_0}{\sum_{m \neq l,k} p_m g_k + x g_k + n_0} \right], \end{aligned} \tag{29}$$

where this last term follows from a similar identity:

$$\frac{b - ax}{c + ax} = -1 + \frac{b + c}{c + ax}, \tag{30}$$

with $a = g_k$, $b = p_0 g_k$, and $c = \sum_{m \neq l,k} p_m g_k + n_0$. The function in (30) is also convex for $x \geq -c/a$, as shown in the visual plot below. This condition on x is trivially true since the powers cannot be negative. Thus, (29) is convex with respect to x.

The summation of convex and constant terms gives a result that is also convex, and thus $f(x)$ is convex. Since a convex function is always maximized at



one of its boundary points, $f(x) = f(p_l)$ is largest either at $p_l = 0, p_k = p_0$, or at $p_l = p_0$, $p_k = 0$.

Thus, in the end, the BS only transmits to one user. CDMA is effectively not used in a single cell!

In general, for an ad-hoc network, if CDMA is used and the rate is a linear function of the SINR, then the throughput-optimal schedule will correspond to the following:

> Each node either transmits to one receiver at its maximum power, or does not transmit to any receivers at all.

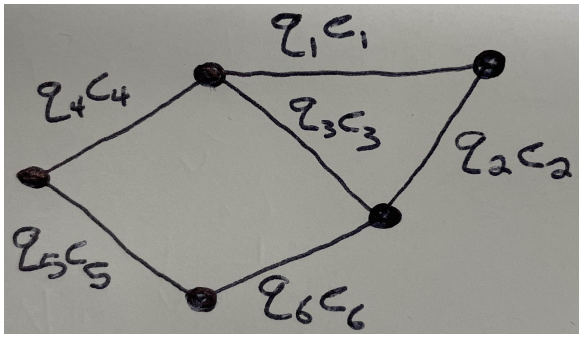
However, the receiver may still receive from multiple transmitters. Determining the set of active transmitters in an ad-hoc network may still be an NP-hard problem in general.

Node-Exclusive Interference Model

In any wireless network, the feasible action space is going to be limited by the interference model. In the cellular case, as we saw above, we can apply our scheduling protocol on top of the multiple access scheme.

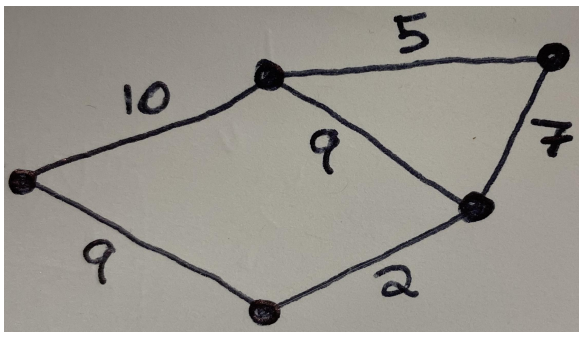
For a more general wireless topology, one possibility is to employ the collision model that we studied previously. In this model, we had limited the actions such that each node has at most one active transmitting/receiving link to avoid interference. We can look at applying our scheduling protocol on top of this, which effectively limits which links can be active.

In the **node-exclusive interference model**, we assume each link l has a fixed capacity c_l if there is no interference. We can construct a weighted graph, where each link weight is the product $q_l(t)c_l$ at time t, such as the following in the case of $N = 5$ nodes and $L = 6$ links:



In this model, each set of non-interfering links corresponds to a graph **matching**. For the **1-hop interference** case – in which nodes that are separated by at least one neighboring node are considered to not be interfering – a matching is a set of links in which no node appears more than once.

The throughput-optimal policy then corresponds to the **maximum-weighted matching** on this graph. This is the matching which has the maximum sum of link weights, which for our scheduling scheme is the product of queue length and capacity. In general, this is non-trivial to compute. For example, consider the following graph:



The maximum-weighted matching is $9 + 9 = 18$, which does not consist of the maximum link weight (10). Algorithms exist to solve this problem in polynomial time, with complexity $O(N^3)$, where N is the number of nodes in the graph. Polynomial time is in general a very desirable property of algorithms, but still, N^3 grows quite fast with the number of nodes!

Thus, an active area of research has always been to find simpler scheduling algorithms with lower complexity. In particular, we may consider *distributed* scheduling, where nodes make decisions locally without having global knowl-

edge. The tradeoff, however, is that we may only be able to support a reduced capacity region.

In some cases, 1-hop interference may not be strict enough – we may need active links separated by more than one neighbor. This can happen if we have nodes operating in close proximity, or if we have very low tolerances on interference. However, once we extend to the **2-hop interference** model, the complexity grows much larger. The throughput-optimal policy is in general NP-hard to find.

Critique

Clearly, our throughput-optimal scheduling algorithm is applicable to a wide range of wireless networks. Still, there are some limitations that are important to keep in mind:

1. We said that one if the nice properties of (3) is that it only needs the current channel state $K(t)$. However, even this may be unknown, or difficult to estimate globally across the network.
2. The policy does not capture delay. The time that particular packets spend in queues can still be large for some throughput-optimal schemes.
3. The policy does not account for hop-by-hop dynamics.
4. Solving the optimization in each timeslot adds a lot of complexity.
5. Other policies may also be throughput-optimal, and could potentially have other desirable properties.

For the max-weight policy, there are a few potential solutions to the complexity issue. One possibility is to solve the max-weight decision approximately. Also, we could consider applying max-weight only to part of the decisions that can be solved by low complexity (e.g., only inside a single cell, with channel assignments chosen beforehand). Finally, we can delegate the more complex decisions to a longer timescale.