Cross-Layer Wireless Design

In this module, so far, we have been focusing on the medium access control (MAC) scheduling problem. However, the capacity region problem is not the only one of interest for wireless data systems.

(1) Routing: How to find good paths through the network for each flow s? We would like to fully use the available capacity, while avoiding any "hot-spots" of congestion.

Often, using a single path cannot achieve the best performance (which can be measured in different ways, e.g., throughput). This leads to the concept of multi-path routing, where multiple paths are utilized for a single flow.

(2) Rate Control: Suppose the rate vector $\vec{X}$ across flows is not given. Rather, the users can adapt their rates to achieve certain objectives. A few points are clear:

- The rates must lie within the capacity region Λ of the network. We want them as large as possible, but need congestion control to ensure we stay within Λ.
- Fairness across the flows must be considered.

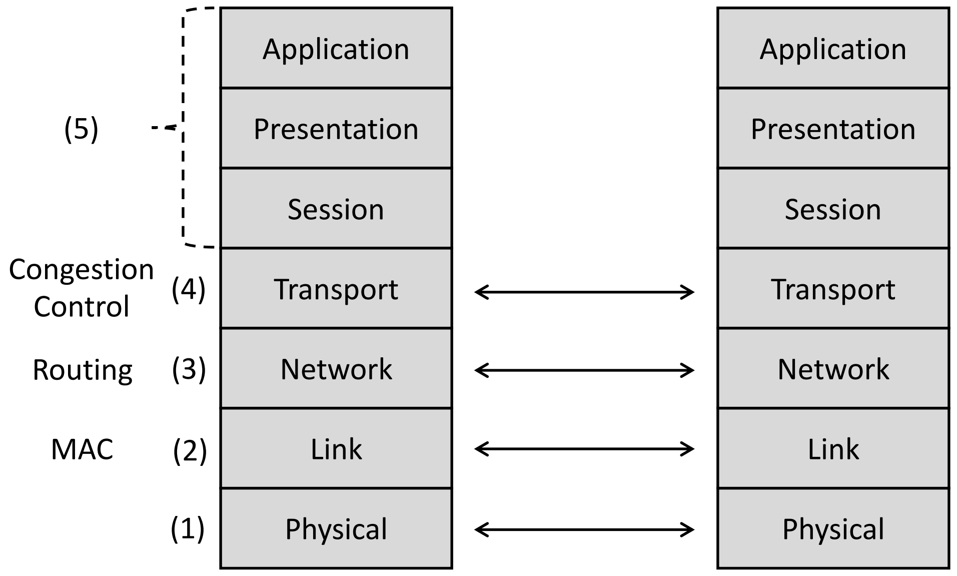
(3) Energy Conservation: We saw that the throughput-optimal scheduling algorithm can stabilize the system for any $\vec{X} \in \Lambda$. However, it may not lead to the policy that has the lowest energy consumption across nodes. Remember that we are dealing with wireless transmitters (e.g., mobiles, base stations), so energy conservation is important.

Often, these three problems (and related ones) are closely tied to the capacity region of the network. As a result, it can be beneficial to design the solution jointly considering these functionalities together with MAC scheduling. This leads to the concept of **cross-layer design**, as it involves joining variables that sit at different "layers" of the network protocol stack.

Motivating Cross-Layer Design

In wireline networks, often the protocols are classified into **layers**. Layering is a form of hierarchical modularity, organizing different functions of the network. The layer above uses the service provided by the layer below, but it does not need to know the inner workings of that lower layer.

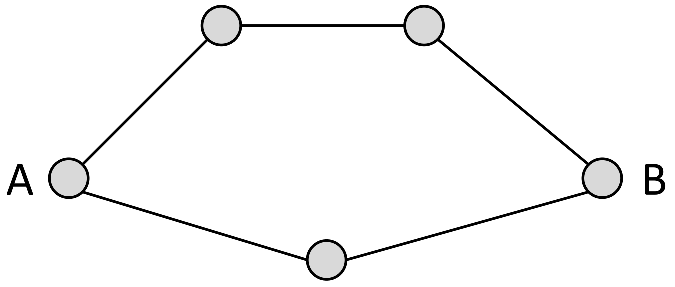
We typically think of the protocol stack as having between 5 and 7 layers (with the top three often being grouped into a single layer):



MAC, routing, and congestion control are relegated to the link, network, and transport layers, respectively. These are layers 2, 3, and 4. The physical layer (layer 1) deals with the physical bits being transmitted. The application layer (layer 5) is the closest to the end user, and the applications being run over the network.

Modularity has several benefits. It is easy to understand. It also makes it easy to swap out the design of a certain layer without impacting the other layers. However, for wireless networks, examples have been found where such a layering architecture can limit performance.

A key example of such a limitation occurs in routing. Typically, routing is designed to minimize the number of hops. In routing from A to B below, the bottom path has less hops and may be chosen as a result:



Fewer hops also means that we are tending to use "long" links. But in wireless networks, long transmission links can suffer from a low SINR, resulting in poor end-to-end performance. Thus, it would be better if the routing protocol (at layer 3) would take into account these physical-layer characteristics (at layers 1&2).

However, cross-layer design also has some pitfalls. We have a loss of modularity once we start integrating functionality across layers. This creates a fragile solution that can be hard to change!

Nonetheless, the Lyapunov-based approach allows us to easily integrate some functionality together. This is due to the fact that many of the solutions consist of a MAC component of the form

$$\max_{\substack{\vec{p}\in\mathcal{H} \\ \vec{r}=g(\vec{p},K(t))}} \sum_l q_l(t) \cdot r_l. \tag{1}$$

The length of the queue q_l becomes the "glue" across layers.

We will look through a few cases of this in the following.

Joint Scheduling and Congestion Control

Suppose that, in addition to optimizing throughput, we want to choose the arrival rate X_s of each flow s to maximize some system objective.

Such objectives are commonly captured through a **utility function**, $U(X_s)$. $U(X_s)$ will be always increasing (or at least non-decreasing) in X_s, and will be **concave** in X_s to capture diminishing marginal returns. A commonly employed utility function is **logarithmic utility**, i.e.,

$$U(X_s) = \log(X_s), \tag{2}$$

which helps to enforce fairness in the sense that no flow's rate will be too large or too small (increasing already large X_s has little improvement to the total utility). More precisely, log-utility corresponds to a type of fairness known as **proportional fairness**.

Formally, we want to consider the following problem:

$$\begin{aligned} &\underset{X_s}{\text{maximize}} \quad \sum_s U(X_s) \\ &\text{subject to} \quad \Big[\sum_s H_s^l X_s\Big] \in \Lambda = \sum_k \lambda_k \cdot \mathsf{Conv\_hull}\{g(\vec{p},k) \mid \vec{p} \in \mathcal{H}\} \end{aligned} \tag{3}$$

Let us begin with the same Lyapunov function as before:

$$V(\vec{q}) = \frac{1}{2}\sum_l q_l^2. \tag{4}$$

Expressing the queue length in terms of X_s, we have

$$\begin{aligned} (q_l(t+1))^2 &\le \big(q_l(t) + \sum_s H_s^l X_s - r_l(t)\big)^2 \\ &= (q_l(t))^2 + 2q_l(t)\Big[\sum_s H_s^l X_s - r_l(t)\Big] \\ &\qquad + \Big[\sum_s H_s^l X_s - r_l(t)\Big]^2. \end{aligned} \tag{5}$$

Since X_s is finite and $r_l(t)$ is bounded, there exists a constant M_l such that

$$(q_l(t+1))^2 \le (q_l(t))^2 + 2q_l(t)\Big[\sum_s H_s^l A_s(t) - r_l(t)\Big] + M_l^2. \tag{6}$$

Thus,

$$V(\vec{q}(t+1)) \le V(\vec{q}(t)) + \sum_l q_l(t)\Big[\sum_s H_s^l A_s(t) - r_l(t)\Big] + \frac{1}{2}\sum_l M_l^2, \tag{7}$$

which implies that

$$\begin{aligned} \mathbb{E}\big[V(\vec{q}(t+1)) - V(\vec{q}(t)) \mid \vec{q}(t)\big] \le& \sum_l q_l(t)\Big[\sum_s H_s^l X_s\Big] - \sum_l q_l(t)\mathbb{E}\Big[r_l(t) \mid q_l(t)\Big] \\ &+ \frac{1}{2}\sum_l M_l^2. \end{aligned} \tag{8}$$

This is the same form we have seen previously, and gives us the drift term. However, maximizing $\sum_l q_l(t)\mathbb{E}[r_l(t) \mid q_l(t)]$ does not help us to choose X_s. To accomplish this, we add an explicit **penalty term**

$$\Delta(t) = \sum_s U(X_s^\star) - \sum_s U(X_s) \tag{9}$$

to (8), where $[X_s^\star]$ is the optimal solution to (3). (9) quantifies how sub-optimal $[X_s]$ is. Including a factor $\alpha \geq 0$ weighting the importance of the penalty term vs. the drift, we have

$$\begin{aligned}\mathbb{E}\big[V(\vec{q}(t+1)) - V(\vec{q}(t)) \mid \vec{q}(t)\big] + \alpha\Delta(t) \leq \underbrace{\sum_l q_l(t)\Big[\sum_s H_s^l X_s\Big] - \alpha \sum_s U(X_s)}_{\text{(a)}} \\ - \underbrace{\sum_l q_l(t)\mathbb{E}\big[r_l(t) \mid q_l(t)\big]}_{\text{(b)}} + \frac{1}{2}\sum_l M_l^2 + \alpha\sum_s U(X_s^\star), \qquad (10)\end{aligned}$$

from which we will try to minimize the drift-plus-penalty. We focus on terms (a) and (b) since the remainder are constant with respect to our variables:

Penalty term (a): Minimizing (a) is the same as maximizing its negative:

$$\begin{aligned} & \max_{[X_s]} \ \alpha\sum_s U(X_s) - \sum_l q_l(t)\Big[\sum_s H_s^l X_s\Big] \\ = \ & \max_{[X_s]} \sum_s \Big(\alpha U(X_s) - X_s \sum_l H_s^l q_l(t)\Big) \\ = \ & \sum_s \max_{[X_s]}\Big(\alpha U(X_s) - X_s \sum_l H_s^l q_l(t)\Big). \end{aligned} \tag{11}$$

This last step of interchanging the summation and maximization is critical because it implies that the problem can be solved in a distributed manner across the flows. Specifically, each flow can choose X_s at time t according to:

$$X_s(t) = \arg\max_{X_s}\Big(\alpha U(X_s) - X_s \sum_l H_s^l q_l(t)\Big). \tag{12}$$

We may interpret $q_l(t)/\alpha$ as the "price" of link l at time t. The larger the queue is, the more it costs to transmit over the link, which is a key concept in **congestion control** protocols traditionally executed at layer 4 of the protocol stack. Further, $\sum_l H_s^l q_l(t)/\alpha$ is the total price along the path of flow s. $X_s(t)$ is thus chosen to maximize the **net utility**: the larger α is, the less emphasis is placed on the congestion over the links relative to the utilities experienced by user flows.

For example, consider the case of log utility in (2). Taking the derivative with respect to X_s and setting to 0, we have

$$\frac{\alpha}{X_s} = \sum_l H_s^l q_l(t) \to X_s = \frac{\alpha}{\sum_l H_s^l q_l(t)}. \tag{13}$$

Drift term (b): Maximizing (b) in (10) leads to the same max-weight scheduling decision that we have already studied. So, we will choose the action $\vec{p}(t) \in \mathcal{H}$ that maximizes $\sum_l q_l(t) \cdot r_l$, where $\vec{r} = g(\vec{p}, K(t))$. Over time, we can keep alternating between (a) optimizing the flow rates in (12) and (b) the link actions to support these rates and maximize the throughput.

Why does this work? Suppose that we know the optimal arrival rates and have chosen them accordingly, i.e., $X_s(t) = X_s^\star$. Then, from (8), we must have

$$\mathbb{E}\big[V(\vec{q}(t+1)) - V(\vec{q}(t)) \mid \vec{q}(t)\big] \le \frac{1}{2}\sum_l M_l^2, \tag{14}$$

since $[X_s^\star] \in \Lambda$ and for any $\vec{X} \in \Lambda$, $\sum_l q_l(t)\big[\sum_s H_s^l X_s\big] \le \sum_l q_l(t)\mathbb{E}\big[r_l(t) \mid q_l(t)\big]$ under our throughput-optimal scheduling policy. Further, $\Delta(t) = 0$ at this point by definition. Together, then, the total drift and penalty in (10) would satisfy

$$\mathbb{E}\big[V(\vec{q}(t+1)) - V(\vec{q}(t)) \mid \vec{q}(t)\big] + \alpha\Delta(t) \le \frac{1}{2}\sum_l M_l^2. \tag{15}$$

Now, since our choice of $X_s(t)$ and $r_l(t)$ from optimizing (10) are meant to minimize the drift-plus-penalty, the upper bound in (15) must be true for decisions resulting from our algorithm (i.e., with potentially non-zero $\Delta(t)$). Summing (15) over time periods $t = 1, ..., T$, and dividing by T, we have

$$\begin{aligned}\frac{\mathbb{E}\big[V(\vec{q}(t+1))\big] - \mathbb{E}\big[V(\vec{q}(0))\big]}{T} &+ \alpha\sum_s U(X_s^\star) - \alpha\frac{1}{T}\sum_{t=1}^{T}\sum_s \mathbb{E}\big[U(X_s(t))\big] \\ &\le \frac{1}{2}\sum_l M_l^2 = M. \end{aligned}\tag{16}$$

Thus, we can lower bound the average utility achieved over time as

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{s}\mathbb{E}\big[U(X_s(t))\big] \geq \sum_{s} U(X_s^\star) - \underbrace{\frac{M}{\alpha}}_{\text{(a)}} + \underbrace{\frac{\mathbb{E}\big[V(\vec{q}(t+1))\big]}{\alpha T}}_{\text{(b)}} - \underbrace{\frac{\mathbb{E}\big[V(\vec{q}(0))\big]}{\alpha T}}_{\text{(c)}}. \tag{17}$$

Note that as $T \to \infty$, terms (b) and (c) approach 0. Thus, over time, the achieved utility is smaller than the optimal $\sum_s U(X_s^\star)$ by at most M/α. Further, as $\alpha \to \infty$, terms (a)-(c) all approach 0, and thus loss in the average utility approaches 0. This is what we would expect when increasing priority is placed on the penalty term as opposed to the drift term.

This emphasizes a *queue vs. optimality* tradeoff. As $\alpha \to \infty$, we expect that $X_s(t)$ becomes close to $X_s^\star$. From (12),

$$\max\Big(\alpha U(X_s) - X_s \sum_{l} H_s^l q_l(t)\Big) \quad \to \quad \alpha U'(X_s) = \sum_{l} H_s^l q_l(t). \tag{18}$$

As α increases, $q_l(t)$ increases. Closer optimality is achieved at the cost of higher queue length.

We can also readily replace the utility measures $U(X_s)$ with other objectives. With energy consumption, for example, we would be aiming to minimize rather than maximize. You will be looking at this in a homework problem.

Joint Routing and Scheduling

The previous section considered joint congestion control (layer 4) and scheduling (layer 2). We will now consider joining scheduling with routing, which is traditionally a layer 3 protocol.

In this model, we have multiple **commodities** $c = 1, ..., C$ to be delivered over the wireless network of nodes $i = 1, ..., N$ and links $l = 1, ..., L$. A commodity can be thought of simply as data relevant to a specific application. At layer 3, the blocks of data transmitted (i.e., for each commodity) are called **packets**.

- Each commodity c has a destination node $d(c) \in \{1, ..., N\}$, which is one of the N nodes.
- A number of the nodes may generate packets of commodity c. We define $\lambda_i^c \geq 0$ as the rate of new packets of commodity c generated by node i. Still, no routes for the commodities have been specified yet.

Further, our link model will be the same as from the scheduling problem: $\vec{r} = g(\vec{p}, K(t))$ for $p \in \mathcal{H}$, with the channel state $K(t)$ being i.i.d. over time. We will find it convenient here to represent a link l in terms of its transmitting node i and receiving node j, i.e., link ij.

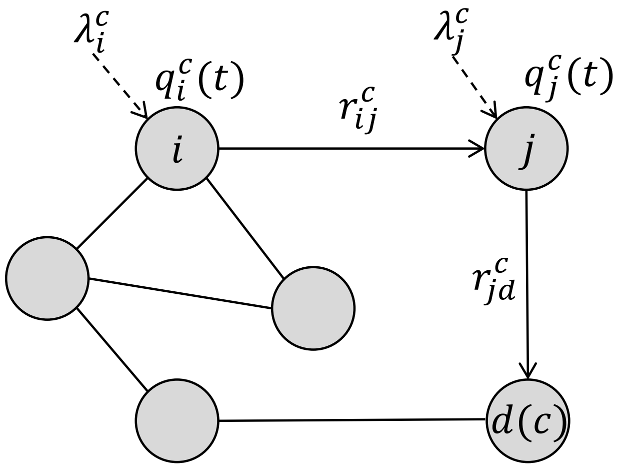
In order for the arrival rates $[\lambda_i^c]$ (across nodes and commodities) to be supported, there must exist a collection of rates $[r_{ij}^c]$ (across links and commodities) such that:

- $r_{ij}^c \geq 0$ is the rate that node i forwards commodity c to node j.
- $\lambda_i^c + \sum_{j \neq i} r_{ji}^c \leq \sum_{j \neq i} r_{ij}^c$ for all nodes $i \neq d(c)$. In other words, a node cannot be transmitting a larger rate than it is generating and receiving.
- $\left[\sum_c r_{ij}^c\right] \in \Lambda = \sum_k \lambda_k \cdot \mathsf{Conv\_hull}\{g(\vec{p}, k) \mid \vec{p} \in \mathcal{H}\}$. In other words, the sum of rates across commodities on each link must be in the capacity region. (Note λ_k, as written here, is distinct from λ_i^c.)

Each node maintains multiple queues, one for each destination/commodity c, which we denote as q_i^c. The queue evolution is thus described as

$$q_i^c(t+1) = \begin{cases} \left[q_i^c(t) + \sum_{j \neq i} r_{ji}^c + \lambda_i^c - \sum_{j \neq i} r_{ij}^c\right]^+ & \text{if } i \neq d(c) \\ 0 & \text{if } i = d(c) \end{cases}. \tag{19}$$

Our overall system model can thus be depicted as follows:



We will now see how working with the Lyapunov function also produces a joint routing and scheduling policy that is throughput-optimal. We will use the Lyapunov function

$$V(\vec{q}) = \frac{1}{2}\sum_{i,c} \left(q_i^c\right)^2. \tag{20}$$

Note that

$$\left[q_i^c(t+1)\right]^2 - \left[q_i^c(t)\right]^2 \leq 2q_i^c(t)\Big[\sum_{j\neq i} r_{ji}^c + \lambda_i^c - \sum_{j\neq i} r_{ij}^c\Big] + \left(M_i^c\right)^2, \tag{21}$$

where M_i is independent of the queue length. Thus, we have

$$\begin{aligned} V(\vec{q}(t+1)) - V(\vec{q}(t)) &\leq \sum_{i,c} q_i^c(t)\Big[\sum_{j\neq i} r_{ji}^c + \lambda_i^c - \sum_{j\neq i} r_{ij}^c\Big] + \frac{1}{2}\sum_{i,c}\left(M_i^c\right)^2 \\ &= \sum_{i,c} q_i^c(t)\cdot\lambda_i^c - \underbrace{\sum_{(i,j)\in\mathcal{L},c} r_{ij}^c\left[q_i^c(t) - q_j^c(t)\right]}_{\text{(a)}} + \frac{1}{2}\sum_{i,c}\left(M_i^c\right)^2, \end{aligned} \tag{22}$$

where $(i,j)\in\mathcal{L}$ denotes a link in the graph. Term (a) comes about from the following sequence of steps, allowing us to interchange the order of $q_i^c(t)$ and r_{ij}^c in the summation:

$$\begin{aligned} \sum_{i,c} q_i^c(t)\sum_{j\neq i}\left(r_{ji}^c - r_{ij}^c\right) &= \sum_{(i,j)\in\mathcal{L},c} q_i^c(t) r_{ji}^c - \sum_{(i,j)\in\mathcal{L},c} q_i^c(t) r_{ij}^c \\ &= \sum_{(i,j)\in\mathcal{L},c} q_j^c(t) r_{ij}^c - \sum_{(i,j)\in\mathcal{L},c} q_i^c(t) r_{ij}^c \\ &= \sum_{(i,j)\in\mathcal{L},c} r_{ij}^c\left[q_i^c(t) - q_j^c(t)\right]. \end{aligned}$$

As in our previous throughput-optimal scheduling analysis, we should maximize term (a) to minimize the drift. At time t, the queues $q_i^c(t)$ are constant, so this term is a weighted sum of the variables r_{ij}^c. For link (i,j), then, we want to use all of its available rate to service the commodity c with the largest differential $q_i^c(t) - q_j^c(t)$, as this will lead to the largest gain in term (a). Letting $r_{ij} = \sum_c r_{ij}^c$, then, term (a) in (22) becomes

$$\sum_{(i,j)\in\mathcal{L}} r_{ij}\cdot\max_c\left(q_i^c(t) - q_j^c(t)\right), \tag{23}$$

and we should choose $\vec{p}(t)$ to maximize this weighted sum.

We can thus design our algorithm to consist of three steps at time t:

1. *Maximum differential backlog:* For each link $(i,j) \in \mathcal{L}$, select the commodity c such that the value $q_i^c(t) - q_j^c(t)$ is the largest. Let $c_{ij}^\star(t) = \arg\max_c (q_i^c(t) - q_j^c(t))$ denote this commodity, and let

$$w_{ij}(t) = q_i^{c_{ij}^\star(t)}(t) - q_j^{c_{ij}^\star(t)}(t) \tag{24}$$

   denote the maximum differential backlog for link (i,j).
2. *Scheduling:* Choose $\vec{p}(t)$ such that

$$\vec{p}(t) = \underset{\substack{\vec{p}\in\mathcal{H} \\ \vec{r}=g(\vec{p},K(t))}}{\arg\max} \sum_{(i,j)\in\mathcal{L}} w_{ij}(t) \cdot r_{ij}, \tag{25}$$

   and let $\vec{r}(t) = g(\vec{p}(t), K(t))$ be the corresponding optimal rate.
3. *Routing:* On each link (i,j), route the commodity $c_{ij}^\star(t)$ using the rate r_{ij}, i.e.,

$$r_{ij}^c(t) = \begin{cases} r_{ij}(t) & \text{if } c = c_{ij}^\star(t) \\ 0 & \text{otherwise} \end{cases}. \tag{26}$$

The intuition here is that as packets are queued, the queue difference forms a "gradient," which points to the optimal direction to forward packets.

It can be shown that this algorithm achieves the largest set of offered loads $[\lambda_i^c]$. (Reference: "Dynamic Power Allocation and Routing for Time Varying Wireless Networks," by Neely, Modiano, and Rohrs, in IEEE INFOCOM 2003.)

Opportunistic Scheduling