Optimization in MANETs

Stephen Boyd

Electrical Engineering Department, Stanford University

Application (users') view

- multiple, competing application resources (bandwidth, latency, error rate, reliability, . . .) provided by network, needed to operate applications at some level of quality
- we use vector $r \in \mathbf{R}^n$ to summarize resources made available to users
- some applications can adapt to varying levels of resources available; others either work or don't
- can be captured by user utility function $U_i(r)$ ($e.g.$, $0/1$ for 'works or not'; $-\infty$ for hard constraint)
- goal: maximize total user utility $U(r) = U_1(r) + \cdots + U_k(r)$ (can include idea of fariness)

Network view

- allocate competing network resources
  (link rates, buffer space, power, time-slot fraction)
  to provide application resources r to applications/users

- can abstract to the achievable region $\mathcal{R}$ of all possible r that can be offered to users

- improvements in hardware, algorithms, coding, protocols, *etc.*, enlarge $\mathcal{R}$ (we hope)

- our problem is to maximize $U(r)$ over $r \in \mathcal{R}$

Some general comments

- r is interface variable between network and users
- it should contain everything needed for the users to determine their happiness/satisfaction levels, $i.e.$, utility
- from user's view, all that matters is $U(r)$
- from network's view, all that matters is whether $r \in \mathcal{R}$ or not

Utility optimization

- how do we maximize $U(r)$ over $r \in \mathcal{R}$?
- this optimization problem can be tractable (convex, possibly after change of coordinates, or otherwise globally solvable) or not
- how do we describe $\mathcal{R}$? for $n = 2$ or 3, we can draw or plot it, but . . .
- let's look at methods where the network doesn't know U, and the users don't know $\mathcal{R}$
- how much information has to pass between users and network to solve the problem (possibly approximately)?

Resource negotiation

- the battleship method
  - users request a particular r^{req}
    (presumably, one that has higher utility than the current r)
  - network either provides r^{req} or says 'sorry, can't do it'
  - *this will take a long time . . .*

- the counter-offer method
  - users request a particular r^{req}
  - network either provides r^{req}, or counter-offer r^{co}
    (hopefully close to r^{req})
  - *better than battleship method*, but network doesn't know U, so the counter-offer is only a guess as to what would please the users

Resource negotiation

- the multiple counter-offer method
  - users request a particular r^{req}
  - network either provides r^{req}, or offers any one of a set of counter-offers $r_1^{\mathrm{co}}, \ldots, r_p^{\mathrm{co}}$
  - *better than counter-offer method* since one of the offers could please users

- the infinite counter-offer method
  - users request a particular r^{req}
  - network either provides r^{req}, or counter-offers a small patch of possible (typically Pareto optimal) resource vectors $\mathcal{R}^{\mathrm{patch}} \subseteq \mathcal{R}$
  - now the users can choose one these ($e.g.$, the one that maximizes U) and (possibly) repeat

Distributed optimization

- this is distributed (layered) optimization, with one interface r
- in some cases, converges to global optimum
- can also have users send a local description of U ($i.e.$, ∇U) to the network, so it can make a good counter-offer

Distributed optimization

what optimization theory tells us:

- battleship and simple offer/counter-offer schemes will be very slow
  ('zeroth order methods')

- if the network makes available a local patch of $\mathcal{R}$, or if the users give a local description of U ($i.e.$, ∇U), fairly simple schemes can work globally in some cases, and well in many
  (primal and dual decomposition)