STATEMENT OF TEACHING PHILOSOPHY

H. B. Acharya

In designing courses, it is necessary to ask, which content is important for the audience, and which needs to be trimmed? However, I believe this question is not sufficient. A course that provides the most relevant content, but in a rigid, encyclopedic manner, serves the student poorly. For example, consider a standard Networks course, where a student is introduced to the OSI stack and told about the data link, network, and transport layers (and a little about the physical and application layers); the student who accepts this model as gospel truth has serious trouble making sense of non-layer concepts such as MPLS or ATM in a more advanced class.

In teaching, my goal is to get the students comfortable with “playing” with the material, seeing it as a work in progress, and perhaps eventually contributing to research to shape it themselves. To achieve this, in addition to the actual information that the student needs to master, I try to bring in context: how and why a certain architecture developed, what the designers were trying to do, and so on. In teaching Networks, for example, in addition to the textbook (Kurose & Ross), I also introduce the principles behind their development (from Patterns in Network Architecture, by John Day) and war stories about their actual history (from Interconnections, by Radia Perlman). In Distributed Systems, I use a textbook (Verissimo & Rodriguez) which presents the process of system design by an architect, and add supplementary material to deal with the styles and principles tried in middleware (Krakowiak) and programming (Varela & Agha). I have not yet delivered the course on Operating Systems, but in helping design the course, I introduced a new textbook (by Anderson & Dahlin), suggested hands-on material (from Operating Systems: Three Easy Pieces, by Arpaci-Dusseau & Arpaci-Dusseau) and helped start a supplementary course on Systems Programming (based on Bryant & O’Hallaron).

My teaching style is simple: I co-opt the students in “performing” the class, so in fact my lectures are all discussion sections. Every new concept is “bookended” by a few minutes of explanation, for introduction and review; the bulk of the class time is spent in several rounds of activity, with one-minute papers to assess how well the material is coming across. For example, in classes such as Introduction to Programming, I pass out several wireless keyboard-touchpads (by Rapoo), which are all connected to the computer driving the display. These circulate around the class. Having set a problem, I then walk around helping as the students talk to each other and edit the code on the screen, trying to complete the whole working program (we use the online IDE, ideone.com). In more advanced classes, such as Distributed Systems, I have a team of students (a different team each day) prepare the lecture with me, and deliver it with me as well. For class projects, the teams are reshuffled using the “jigsaw” method, so each project team has one expert for each topic. Besides strong student approval, such a participative class keeps interest high – the students “own” the class, so they feel free to play with the material and try out ideas; this has led to a dramatic improvement in the quality of project work.