```
type op_kind = enum(op1, ..., opn);
type arg_type = union(arg1, ..., argn);
type result_type = union(res1, ..., resn);
chan request(int clientID, op_kind, arg_type);
chan reply[n](res_type);
process Server {
  int clientID; op_kind kind; arg_type args;
  res_type results; declarations of other variables;
  initialization code;
  while (true) {    ## loop invariant MI
    receive request(clientID, kind, args);
    if (kind == op1)
      { body of op1; }
    ...
    else if (kind == opn)
      { body of opn; }
    send reply[clientID](results);
  }
}
process Client[i = 0 to n-1] {
  arg_type myargs; result_type myresults;
  place value arguments in myargs;
  send request(i, opj, myargs);    # "call" opj
  receive reply[i](myresults);     # wait for reply
}
```

Figure 7.5 Clients and server with multiple operations.