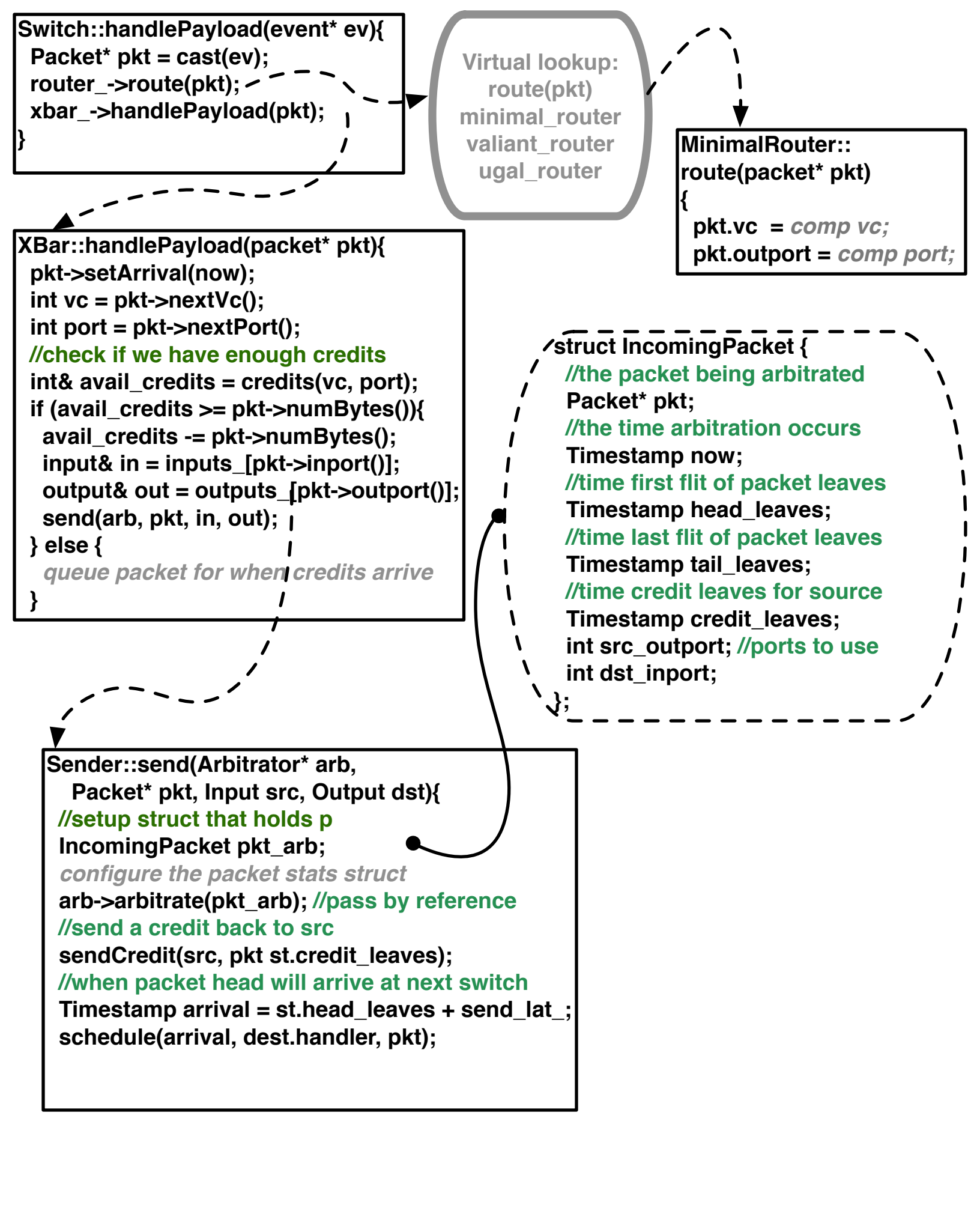
Switch::handlePayload(event* ev){
Packet* pkt = cast(ev);
router_->route(pkt);
xbar_->handlePayload(pkt);
}
Virtual lookup:
route(pkt)
minimal_router
valiant_router
ugal_router
MinimalRouter::
route(packet* pkt)
{
pkt.vc = comp vc;
pkt.outport = comp port;
XBar::handlePayload(packet* pkt){
pkt->setArrival(now);
int vc = pkt->nextVc();
int port = pkt->nextPort();
//check if we have enough credits
int& avail_credits = credits(vc, port);
if (avail_credits >= pkt->numBytes()){
avail_credits -= pkt->numBytes();
input& in = inputs_[pkt->inport()];
output& out = outputs_[pkt->outport()];
send(arb, pkt, in, out);
} else {
queue packet for when credits arrive
}
struct IncomingPacket {
//the packet being arbitrated
Packet* pkt;
//the time arbitration occurs
Timestamp now;
//time first flit of packet leaves
Timestamp head_leaves;
//time last flit of packet leaves
Timestamp tail_leaves;
//time credit leaves for source
Timestamp credit_leaves;
int src_outport; //ports to use
int dst_inport;
};
Sender::send(Arbitrator* arb,
Packet* pkt, Input src, Output dst){
//setup struct that holds p
IncomingPacket pkt_arb;
configure the packet stats struct
arb->arbitrate(pkt_arb); //pass by reference
//send a credit back to src
sendCredit(src, pkt st.credit_leaves);
//when packet head will arrive at next switch
Timestamp arrival = st.head_leaves + send_lat_;
schedule(arrival, dest.handler, pkt);