Dynamic Transmission Power Control (In NS2)

Power Control schemes suitably vary transmission power to reduce energy consumption. In addition to providing energy saving, power control can potentially be used to reduce interference and improve spatial channel reuse. Till now, we have only studied power control for the purpose of energy saving. In the future, we would like to study how to increase network throughput by exploiting it.

1. First, you need to write a function that can calculate the minimum transmission power required to transmit a packet to the desired distance.

You can use the two-ray ground propagation model. In tworayground.cc, add

```
double TwoRayGround::Pt(double dist, double rxThreshold, Antenna* ant, WirelessPhy *ifp) {

  double L = ifp->getL();                    // system loss
  double lambda = ifp->getLambda();       // wavelength

  // assume mobilenode is located at z=0.0 since ns2 doesn't support 3-D plane
  // this is more or less a hack since the receiver's antenna and wireless-phy
  // information are not availible at the sender's side, but it is safe to assume
  // all mobile nodes use the same physical interfaces. This assumes that RXThreshold_
  // and lambda_ value is the same.
  double hr = 0.0 + ant->getZ();
  double ht = hr;
  // Note: for Gt and Gr all availible antenna in ns2 either returns the default
  // value 1.0 or the specified value. Until the omni-antenna is modified. Just
  // give dummy values to getRxGain and getTxGain's parameters
  double Gr = ant->getRxGain(0.0, 0.0, 0.0, lambda);
  double Gt = ant->getTxGain(0.0, 0.0, 0.0, lambda);

  // WHAT A BIG HACK!!!
  // calculate cross over distance
```

```
double tempCrossover_dist = (4 * PI * ht * hr) / lambda;

double Pt = 0.0;
if (dist <= tempCrossover_dist) {
      // use Friis propagation model
      //
      //       (4 * pi * d)^2 * L * Pr
      // Pt = -------------------------
      //        Gt * Gr * (lambda^2)
  Pt = ( ((4 * PI * dist)*(4 * PI * dist) * L * rxThreshold ) /
               (Gt * Gr * lambda * lambda ) );
}
// use two ray ground model
else {
  // ------------------------------------------------------
  // Two Ray ground reflection model according to the equation above
  //
  //               Pr * d^4 * L
  // Pt = ----------------------------------
  //               ( Gt* Gr * (ht^2 * hr^2) )
  Pt =( (rxThreshold * dist * dist * dist * dist * L  ) /
               (Gt * Gr * (ht * ht * hr * hr )) );

}
return Pt;
}
```

2. Then you use your own calculated transmission power to replace the original one in wireless-phy.cc

```
void WirelessPhy::sendDown(Packet *p)
{
    ......
    ......
    txPower = ((TwoRayGround*)propagation_)->Pt(dist, RXThresh_, ant_, this); //get transmission power
    ......
    // p->txinfo_.stamp((MobileNode*)node(), ant_->copy(), Pt_, lambda_);
    // modified this line to use txPower
    p->txinfo_.stamp((MobileNode*)node(), ant_->copy(), txPower, lambda_);
    ......
}
```

3. If you only need to study the network connectivity in topology control, you can finish here. However, if you want to further consider the energy consumption, you should also change the calculation of energy consumption. I have not found a proper energy model to relate the transmission power with the energy consumption. A proper way would be using datasheets of RF radios. However, we cannot expect improvements of energy consumption because currently circuit electronics dominate the energy consumption of low-power RF transceivers.

(1) I tried GRAB's model

```
txPowerConsume = (txPower/Pt_)*Pt_consume_;
```

Obviously, in this model, transmission can consume much lower power than receiving and idle listening. Nodes in transmission mode may save more energy.

(2) Then I resorted to LEACH's simple radio model

```
double hr, ht; // height of recv and xmit antennas
ht = 0.0 + ant_->getZ();
hr = ht; // assume receiving node and antenna at same height
double crossover_dist = (4 * PI * ht * hr) / lambda_;

if (d < crossover_dist) {
   if (d > 1)
      txPowerConsume = Efriis_amp_ * bandwidth_ * d * d;
   else
```

```
        // Pfriis_amp_ is the minimum transmit amplifier power.
        txPowerConsume = Efriis_amp_ * bandwidth_;
    } else {
      txPowerConsume = Etwo_ray_amp_ * bandwidth_ * d * d * d * d;
    }
```

Note: Previously I tried to make the leach's energy model to be consistent with the propagation model so that txPower = txPowerConsume. However, it is more accurate to let it calculate the txPower and txPowerConsume respectively.
As an example,

If txPower is calculated by LEACH's energy model

txPower = Etwo_ray_amp_ * bandwidth_ * d * d * d * d;
= 0.013476e-12 * 1e6 * d^4

and the maximum transmission range is set to 100m, then

Pt_ = txPower = 1.3476 W

Now, to be consistent with the two-ray propagation model, we have to set

RXThresh_ 6.82222e-8 100m
CSThresh_ 2.91229e-9 220m (see tools)

In the two-ray ground model,

Pt = ((rxThreshold * dist * dist * dist * dist * L) / (Gt * Gr * (ht * ht * hr * hr)))
= (6.82222e-8 / 5.0625) * d^4
= 1.347599 * d^4

Although they are similar, considering the calculation error, we have to use the two-ray ground model to calculate the txPower rather than the leach's energy model. NS2 inversely uses the two-ray ground model to calculate the received power Pr_ and the result of leach's model is slightly larger. On the contrary, we can discard leach's energy model and let txPowerConsume = txPower.

Related Scripts: wireless-phy.cc, tworayground.cc, mesh-header.h (In your own defined packet header, you may need to define a variable to specify each packet's transmission range or transmission power so that the wireless layer can learn it).

RETURN