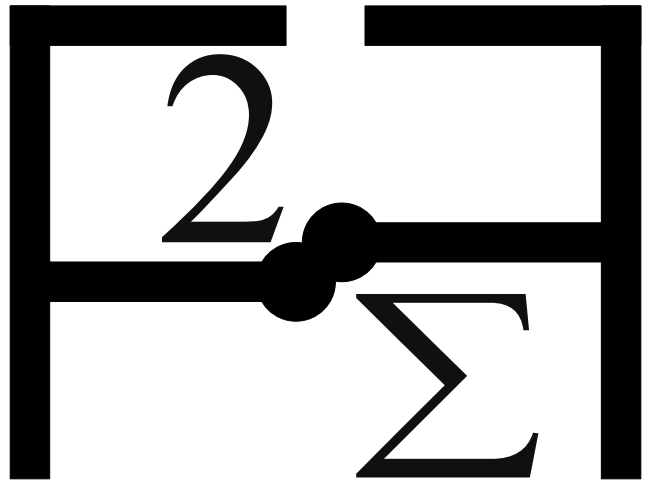
Friend-To-Friend (F2F) Computing



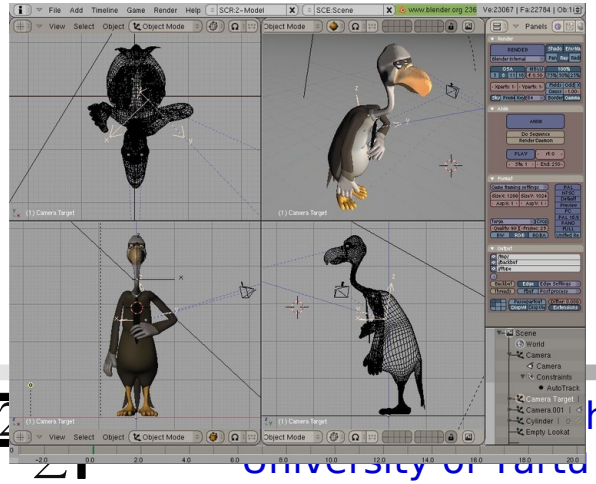
Ulrich Norbisrath

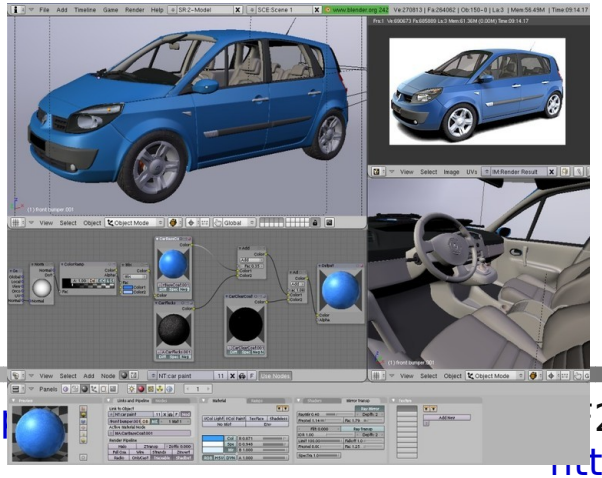
University of Tartu, Estonia

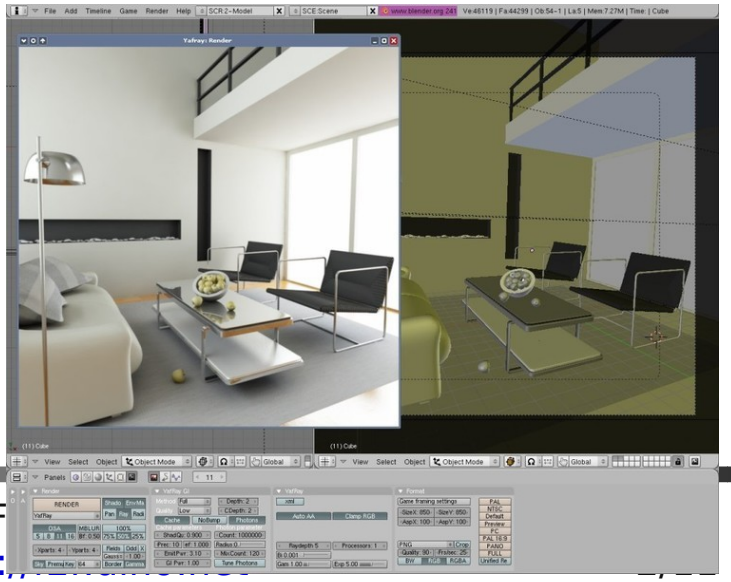
Presentation at
IST'2008 Symposium
2008/10/27

Scenario: Simple Cycles for Small Research Group or Student

- Research group or a student: big computational task (render a movie)
- Users have no Grid certificates/ don't know how to use it or how to submit jobs, want to do it now
- Basic knowledge of a higher level programming language







What can we do?

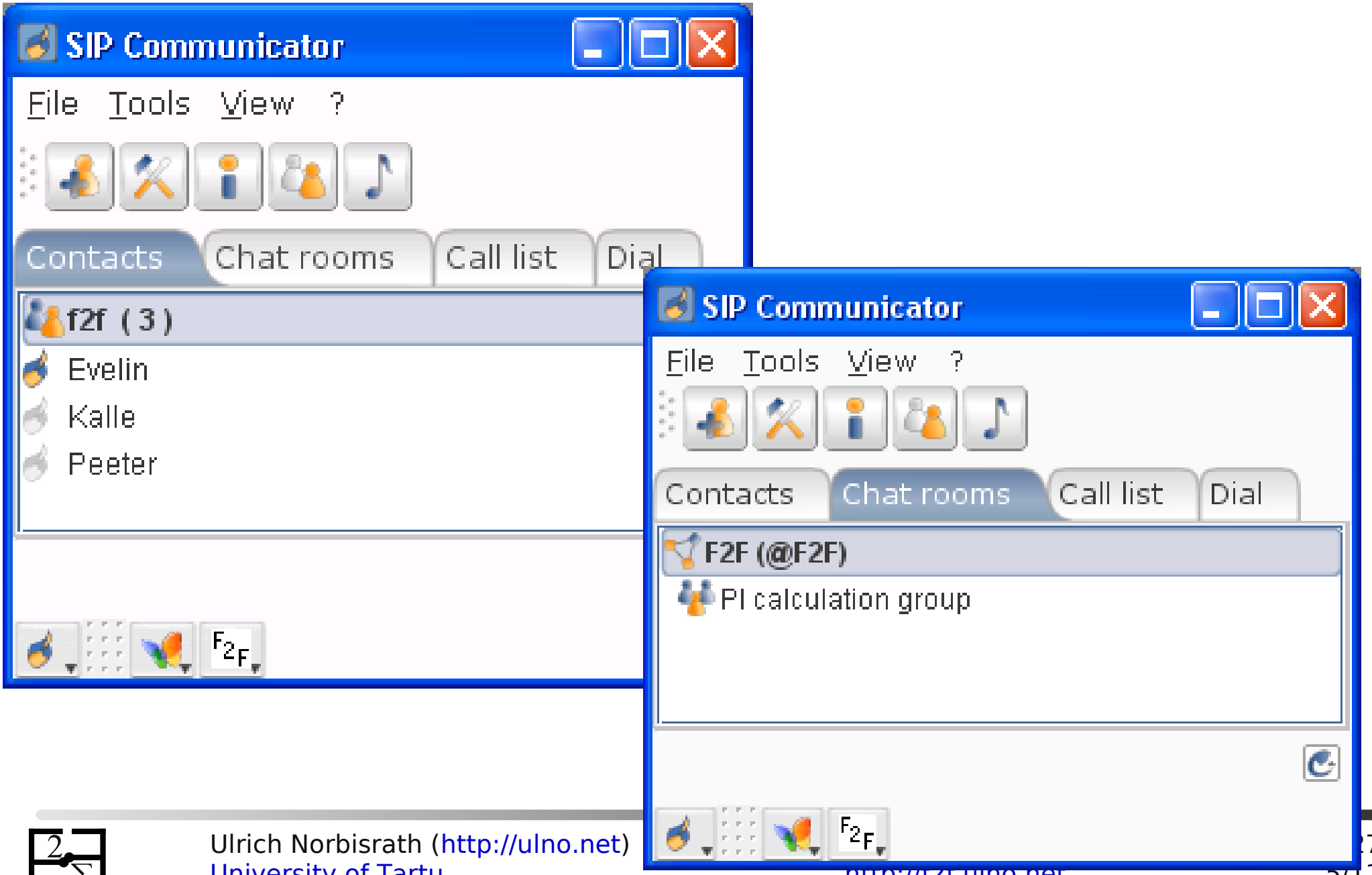
- The (a) grid?
- The cloud?
- Desktop Grid, i.e.: BOINC (seti@home)?

- Why can't we ask our friends/colleagues (with whom we are anyway chatting) for help?
- Why can't the whole thing be as simple as Skype?

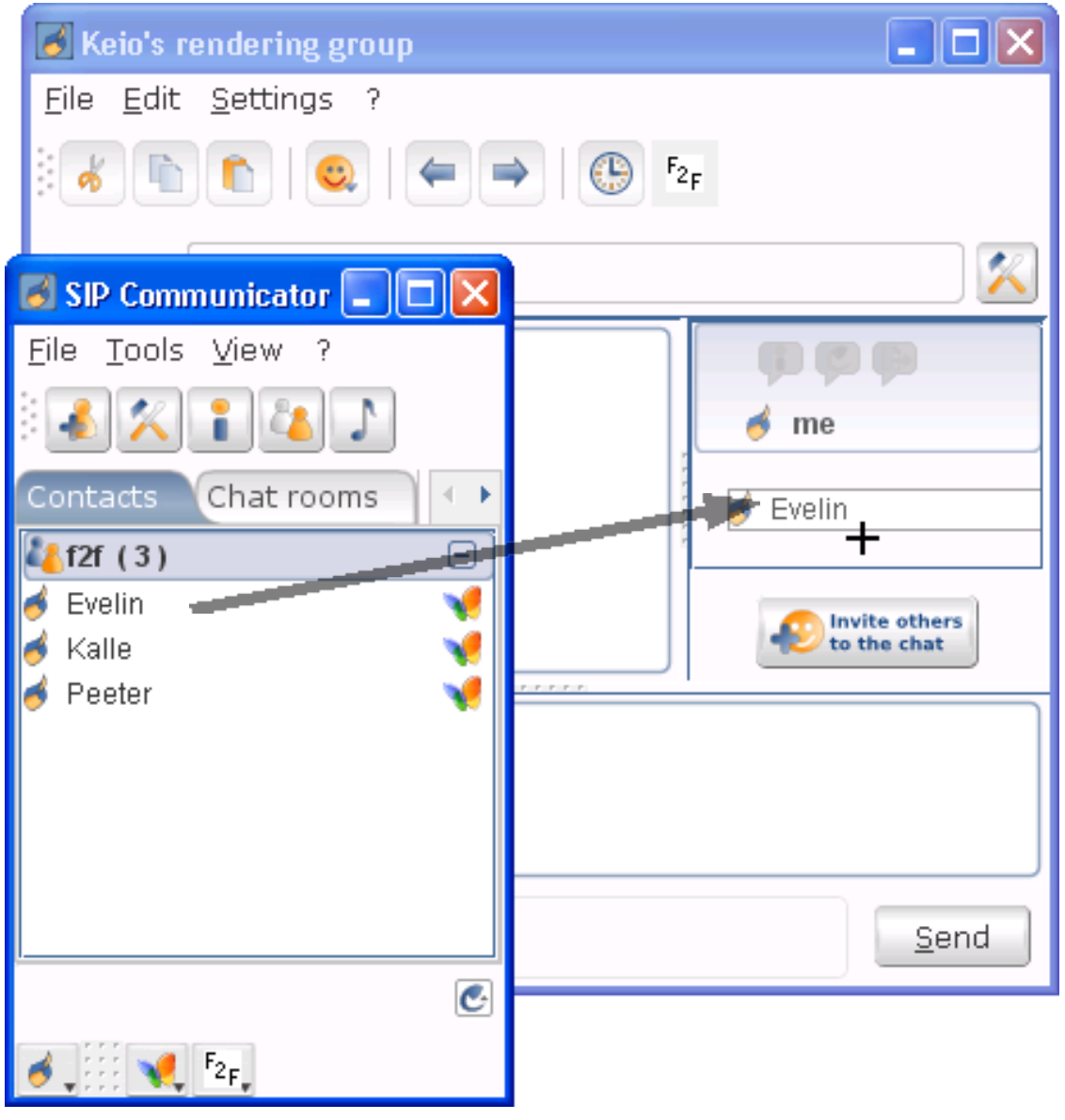
The Frid – The Spontaneous Desktop Grid

- How can
  - we set up a Grid spontaneously without effort?
    - no effort for (need of) administrators
    - no effort for users
  - it be as easy as making a conference chat in Skype or other instant messaging (IM) programs?
- Answer:
  - combine social networks from IM with peer-to-peer (P2P) techniques
    -> Friend-to-Friend Computing -> Frid

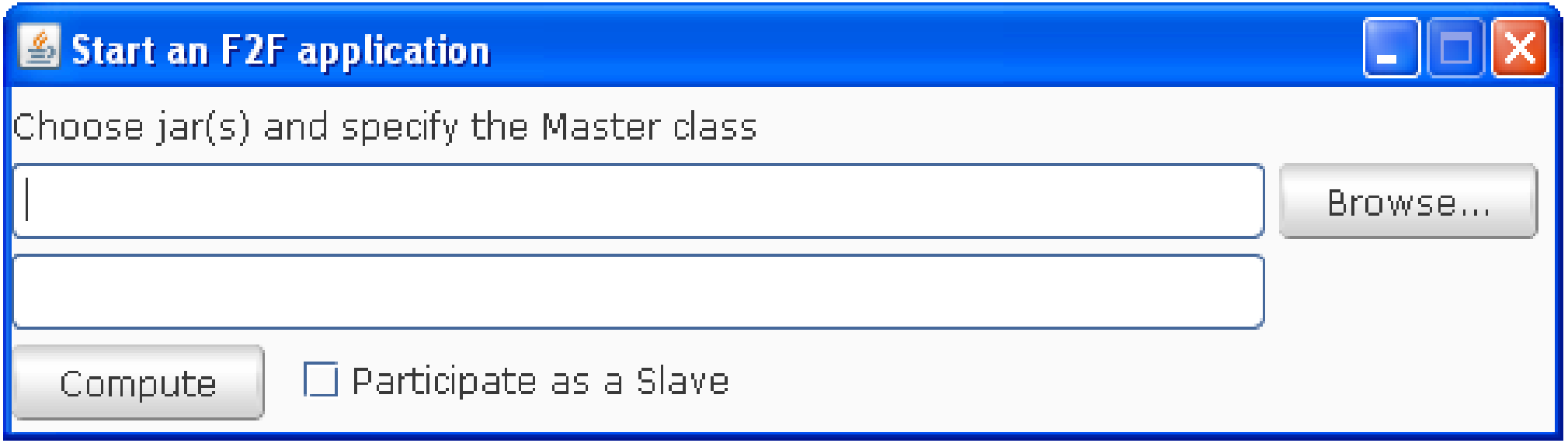
F2F Computing In Practice



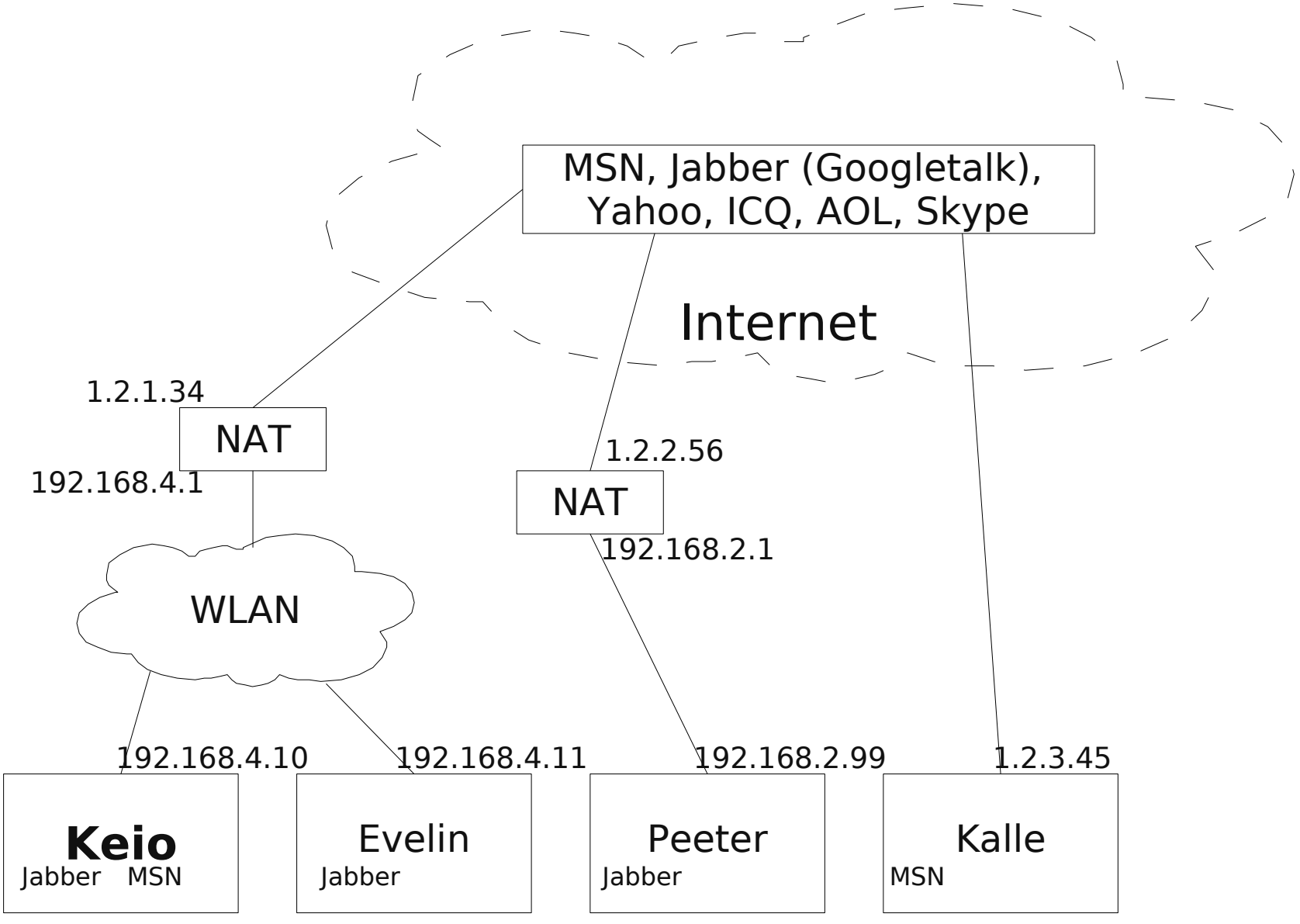
F2F Computing In Practice



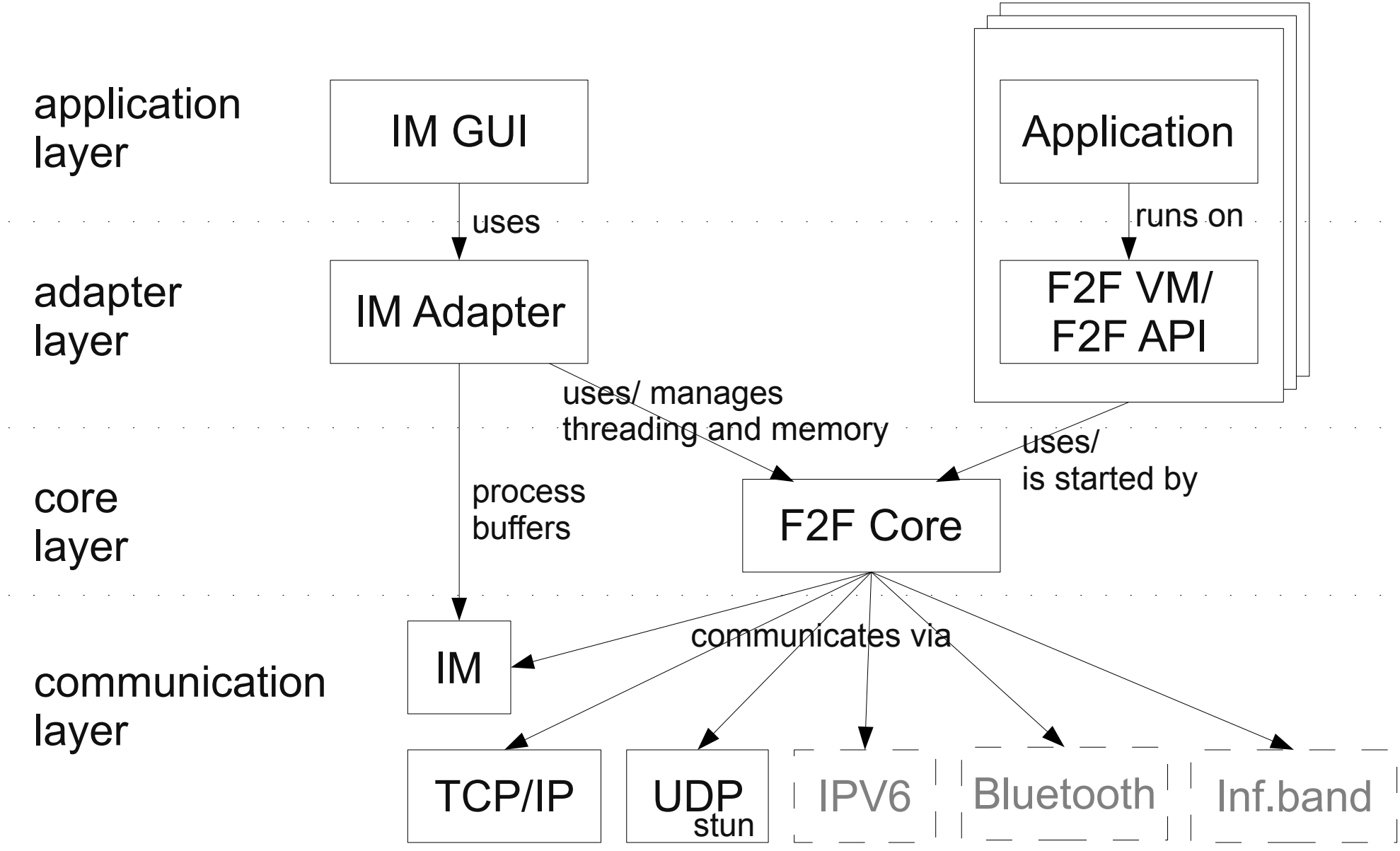
F2F Computing In Practice



The network, we find today



F2F Computing Architecture



Feature List F2F 0.0.1 (06/2008)

- Multi-IM-Protocol support (Skype, MSN, Yahoo, Jabber, ...)
- Real P2P
- UDP NAT traversal
- Java MPI layer
- Released under LGPL at http://f2f.ulno.net
- Pure Java
- Works!!! (Monte Carlo, Blender, MPI Examples)

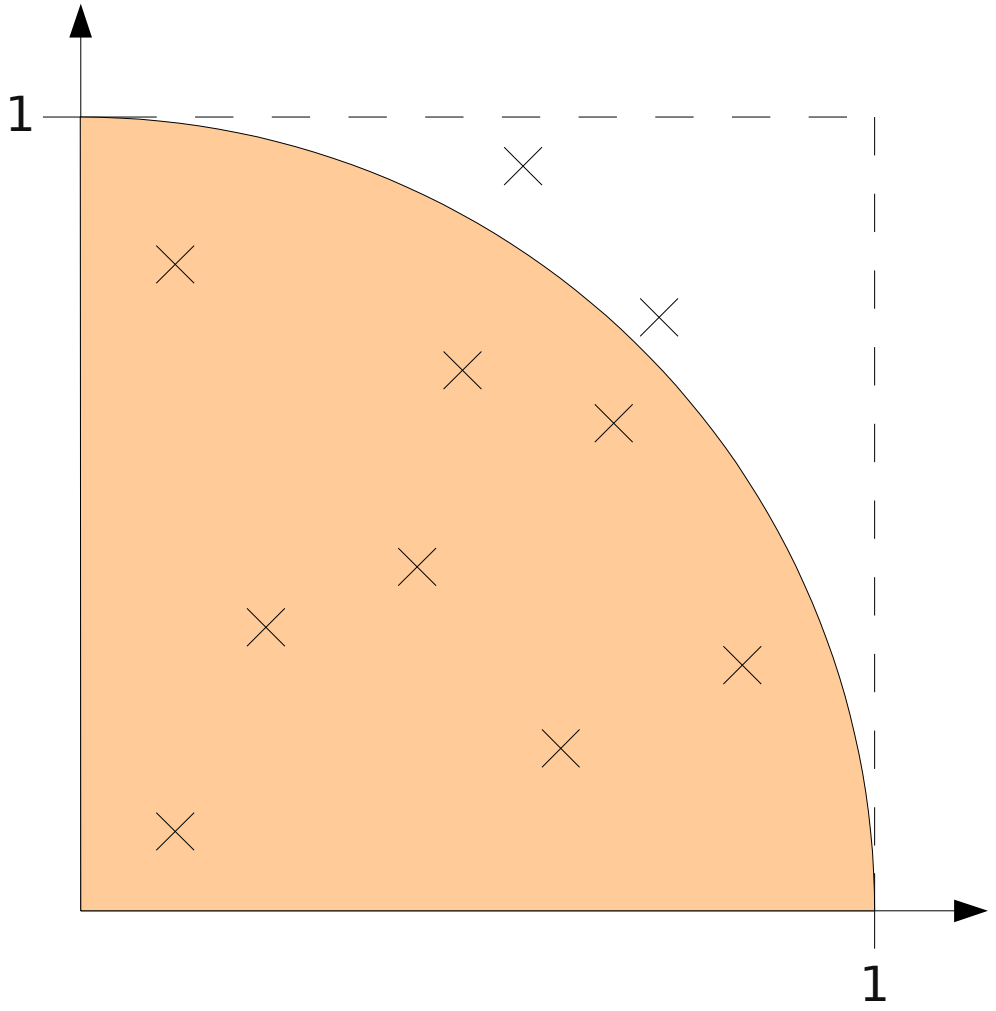
Current projects (F2F2.0)

- General
  - Inheritance of trust (friends of friends)
  - Efficient core (rewritten in C, low footprint) (*)
  - Multi-langage support (python (*), C#, llvm-code)
  - Usage/unification of existing Grids (*)
- Security
  - Encryption
  - Sandboxing (secure javavm, pythonvm, llvm)
- Topology monitoring

(*) initial implemantation ready

Appendix

Monte Carlo Pi (don't try this at home)



- depicted area is $\frac{\pi}{4}$
- choose uniformly distributed random points
- compute average and multiply with 4
  8/10 * 4 = 3.2
- do this >>10 times
- 3.1415926...

Master (1/2)

```
public class PiMasterTask extends Task
{
  public void runTask()
  {
    this.getJob().submitTasks( "ee.ut.f2f.examples.pi.PiSlaveTask",
    this.getJob().getPeers().size(), this.getJob().getPeers());

    Collection<String> taskIDs = this.getJob().getTaskIDs(); // TaskIDs

    // Get Proxies
    Collection<TaskProxy> slaveProxies = new ArrayList<TaskProxy>();
    for (String taskID: taskIDs)
    {
          if (taskID == this.getTaskID()) continue; // not mastertask
          TaskProxy proxy = this.getTaskProxy(taskID);
          if (proxy != null) slaveProxies.add(proxy);
    }
```

Master (2/2)

```
    long intervalms = 2000L; // compute interval
    long maxpoints = 1000000000L; // total points to compute
    AtomicLongVector received = new AtomicLongVector( 0, 0 );

    for (TaskProxy proxy: slaveProxies){ // Send intervaltime
       proxy.sendMessage(new Long(intervalms)); }

    while (received.getUnSyncTotal() < maxpoints)
    {
      Thread.sleep(100);
      for (TaskProxy proxy: slaveProxies) { // check proxies
        while (proxy.hasMessage()) {
          AtomicLongVector receivedvector =
              (AtomicLongVector) proxy.receiveMessage();
          received.add( receivedvector );  } }
    }

    F2FDebug.println("The computed Pi is : " +
      received.getUnSyncPositive() * 4.0 / received.getUnSyncTotal());
  }
}
```

Slave

```
public class PiSlaveTask extends Task {
  AtomicLongVector computedPoints = new AtomicLongVector (0,0);
  MersenneTwisterRNG random = new MersenneTwisterRNG();

  public void runTask() {
    TaskProxy masterProxy = this.getTaskProxy(this.getJob().getMasterTaskID());
    long intervalms = ( (Long)masterProxy.receiveMessage() ).longValue();
    // Start a thread which computes the points
    ComputePoints compTask = new ComputePoints ();
    compTask.start();
    while (true) { // Send the computed points back to the master
      Long stopcondition = (Long) masterProxy.receiveMessage( intervalms );
      if ( stopcondition != null ) if ( stopcondition.longValue() == 0 ) break;
      masterProxy.sendMessage( computedPoints );
      computedPoints.set(0, 0); } //reset
  }

  private class ComputePoints extends Thread{
    public void run() {
       double x = random.nextDouble(); double y = random.nextDouble();
       if( x*x + y*y < 1.0 ) // check if in circle
         computedPoints.positiveHit();
       else computedPoints.negativeHit(); }
  }
}
```