[Bug 954] Support Time::SetResolution in user code

ns-3 Developer Meeting

INRIA, France March 8, 2013

Peter D. Barnes, Jr,

Lawrence Livermore National Laboratory

LLNL-PRES-xxxxxx

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Basic Solution

nstime.h:

```
class Time  {
public:
  inline Time () : m_data ()  {
    TimeSet (this);     }
  }
  ~Time ()  {
    TimeUnset (this);   }
  }

  static void SetResolution (...);
Private:


  typedef ... Times_set;

  static Times_set ** PeekTimeSet ();
  static Times_set * GetTimeSet (...);
  static void DeleteTimeSet ();
  static void TimeSet   (Time *);
  static void TimeUnset (Time *);
};
```

time.cc:

```
Time::Times_set **
Time::PeekTimeSet (...)  {
  static Times_set * times = new Times_set;
  return &times;
}
Time::Times_set *
Time::GetTimeSet (...)  {
  return * PeekTimeSet ();
}
void Time::DeleteTimeSet () {
  Times_set ** times = PeekTimeSet ();
  delete *times;
  *times = 0;
}

void
Time::TimeSet (Time * const time)  {
  Times_set * times = GetTimesSet();
  if (times)  {
    times->insert ( time);
    if (times->size () == 1)  {
      Simulator::Schedule ( Seconds (0), ...);
}}}

void Time::TimeUnset (Time * const time)  {
  Times_set * times = GetTimesSet ();
  if (times)  {
    times->erase (time);
}}
void Time::SetResolution (...) {
  GetTimeSet (deleteMe);
}
```

Simulation Phases

```
# Top of file


int main (int argc, char ** argv)
{



  Simulator::Run ();



  return 0;
}
```

- Static initialization before `main ()`
- User code
- `AtStart ()`
- Simulation running
- User clean up
- Static clean up

Requirements

0. `Time` c'tor/d'tor must be thread-safe!

   `ScheduleWithContext ()`
1. Simulation running
   - Inline Time c'tor, no function calls, no critical sections
2. Static initialization before `main ()`
   - `Time` objects can be constructed before any static code in `time.cc`
3. User code
   - User should be able to `SetResolution ()`
   - Code should fix up any outstanding `Time` objects
4. AtStart
   - Final `SetResolution` (if not already done by user)

Simulation Phases

```
# Top of file


int main (int argc, char ** argv)
{



  Simulator::Run ();



  return 0;
}
```

2. External static init
3. User SetResolution
4. Final Set Resolution
1. No function calls
0. Thread-safety

Alternatives We Considered

- `ScheduleWithContext(…)` call tree has to be thread-safe
- Change to `ScheduleWithContext (uint64_t,…)`
  - Explicit callers have to normalize to current unit
  - Exists in all NetDevices(?), 125 references total
- Add `Time::Unit` to each `Time` object
  - Normalize in operators +-<>=, Scheduler Get…
  - Increased memory and execution time. Significant? Would need to measure.

Implementation Sketch

nstime.h:

```
class Time  {
public:
  inline Time () : m_data ()  {
    if (m_trackResolution)  {
        TimeSet (this);     }
  }
  ~Time ()  {
    if (m_trackResolution)  {
        TimeUnset (this);   }
  }
  static bool GetTrackResolution ();
  static void SetResolution (...);
Private:
  static bool m_trackResolution = true;

  typedef ... Times_set;
  /* CRITICAL SECTIONS */
  static Times_set * GetTimeSet (...);
  static void TimeSet   (Time *);
  static void TimeUnset (Time *);
};
```

1

3

2

0

time.cc:

```
Time::Times_set ** Time::PeekTimeSet () {...
Time::Times_set *  Time::GetTimeSet () {...}
void Times::DeleteTimeSet () {...}

void Time::TimeSet (Time * const time)  {
  TimesSet * times = GetTimesSet();
  if (times)  {
    ret = times->insert ( time);
    if (times->size () == 1)  {
      Simulator::Schedule ( Seconds (0), ...
}}}

void Time::TimeUnset (Time * const time)  {
  TimesSet * times = GetTimesSet ();
  if (times)  {
    times->erase (time);
}}
```

4

Lawrence Livermore
National Laboratory