L14: Time & Ordering

Nickolai Zeldovich
6.033 Spring 2011
Physical oscillators

RC circuits

Quartz crystals

Cesium atoms
Reference clock

US Naval Observatory
Reference clock

US Naval Observatory

WWVB
Simple time sync protocol

csync(server):
    t0 = local_time
    tsrv = getTime(server)
    t1 = local_time
    delay = (t1-t0) / 2
    offset = (t1-delay) - tsrv
    local_time = local_time - offset

gTime():
    return local_time
sync(server):
    t0 = local_time
    tsrv = getTime(server)
    t1 = local_time
    delay = (t1-t0) / 2
    offset = (t1-delay) - tsrv

    freq = base + $\varepsilon$ * sign(offset)
    sleep(freq * abs(offset) / $\varepsilon$)
    freq = base

    { temporarily speed up / slow down local clock }

timer_intr():    # on every oscillator tick..
    local_time = local_time + 1/freq
Adjust frequency estimate

```python
sync_freq(server):
    t0 = local_time
    ts0 = getTime(server)
    sleep(N)
    t1 = local_time
    ts1 = getTime(server)
    ratio = (t1-t0) / (ts1-ts0)
    freq = freq * ratio
```

```python
timer_intr():  # on every oscillator tick..
    local_time = local_time + 1/freq
```
File reconciliation between computers A, B, and C

A

```
abc  def
```

B

```
abc  def
```

C

```
abc  def
```

DEF

---

edit

reconcile
Reconcile based on timestamps

# invoked on candidate file from remote machine
reconcile(remote):
    if local.modTime ≥ remote.modTime:
        return
    local.data = remote.data
    local.modTime = remote.modTime
Reconcile based on timestamps
Problem: concurrent change lost

A's change (def) lost
# invoked on candidate file from remote machine

```
reconcile(remote):
    if local.modTime[n] ≥ remote.modTime[n], ∀n:
        return
    if local.modTime[n] ≤ remote.modTime[n], ∀n:
        local.data = remote.data
        local.modTime = remote.modTime
    else:
        raise ReconcileConflict(local, remote)
```
Vector timestamps

T=(0,0,0)  T=(1,0,0)

abc  def

A

T=(0,0,0)  T=(1,0,0)

abc  def

B

T=(0,0,0)  T=(0,0,5)

abc

C

CONFLICT
Vector timestamps

A

T=(0,0,0)  T=(1,0,0)

abc  →  def

B

T=(0,0,0)  T=(1,0,0)  T=(1,0,5)

abc  →  def  DEF

C

T=(0,0,0)  T=(1,0,0)  T=(1,0,5)

abc  →  def  DEF
Vector timestamps: only local comparison

# invoked on candidate file from remote machine
reconcile(remote):
    if local.modTime[n] ≥ remote.modTime[n], ∀n:
        return
    if local.modTime[n] ≤ remote.modTime[n], ∀n:
        local.data = remote.data
        local.modTime = remote.modTime
    else:
        raise ReconcileConflict(local, remote)