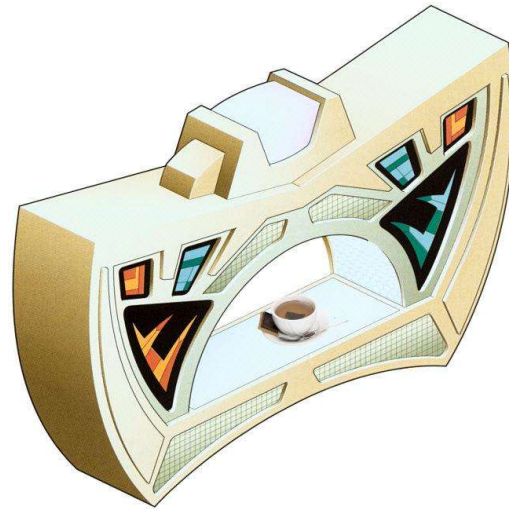


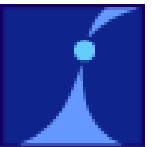
WvSync

(Tea, Earl Grey, Hot:
Replication with WvSync)



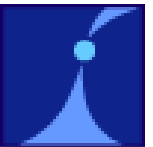
Dave Coombs – Crazy Person
dcoombs@nit.ca

GVADEC – Jun 29, 2004



Outline

- Story Time
- Generalized Replication
- Gremlins
- The Grand Scheme
- Smoke and Mirrors

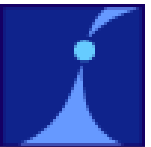


Heckling Policy

- If I say something stupid, tell me.
 - I can take a lot of abuse.

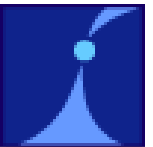


- Free t-shirts for the best hecklers!



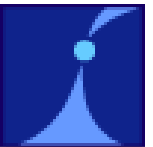
Speed

- Fact: I am hyper.
- Please tell me to slow down.
 - Use force if necessary.



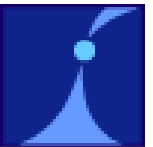
Pronunciation Note

- “Wv-” is pronounced “Weave”
- Not “Wuv”
- Not “Wev”
- **Definitely** not “Double-you-vee”!
- Try it:
 - WvSync == WeaveSync
 - WvDial == WeaveDial



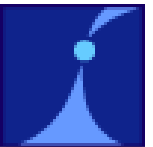
Story Time

- What do these things have in common?
 - backups
 - remote backups
 - groupware
 - user account and DNS synchronization between servers
 - MySQL database replication
 - file synchronization
 - distributed filesystem



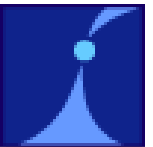
Too Many Replication Projects!

- Separate projects, solving same problems, except:
 - different bugs
 - different design flaws
 - different trade-offs in different places
- Must be a general solution...



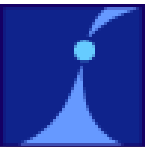
WvSync

- General-purpose replication library
 - C++
 - uses WvStreams and librsync
- Can synchronize “named objects” of arbitrary type
- Sane API
 - Add support for your own arbitrary object type!



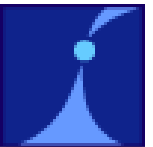
Potential Applications

- File synchronization
 - I like Maildir!
- Desktop synchronization
- Groupware
- Database replication
- “Poor man's” distributed filesystem
- ???



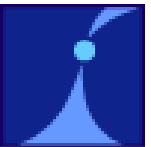
Replication Requirements

- Some data in some format.
 - Big performance / reliability / bandwidth / latency requirements
 - ∴ Copy data, or parts of data, to different places
 - ∴ Must choose which data, how many times, how often, and to where, to replicate
 - Need software to do that :)
-
- Starts to get implementation-dependent at this point



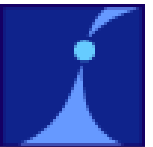
The Usual Compromises

- Result from constraints on **only**:
 - CPU, storage, bandwidth, latency, reliability requirements
- Backup? Waste space to save time.
- Web cache? Waste bandwidth, storage, and reliability to reduce latency.
- What do I make a copy of, and what do I not?
 - Some solutions only do one or the other



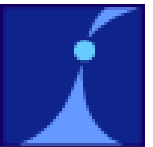
Optimal Compromise

- Where you don't have to choose between “online” and “offline” modes
 - Replicate a “set of named objects”
 - Convert named objects into actual things
 - Hard: Provide API for accessing objects
 - Waste all idle bandwidth
 - Critical/non-critical operations?
 - Negative latency!



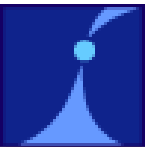
Optimal Comprose II

- Waste all available storage space
 - Cache **everything**
 - Let protocol tell you when cache invalid
- Waste all available CPU time
 - Track what data is used most often
 - Schedule replications based on that
 - Use something like librsync



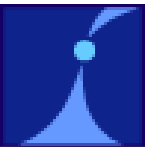
When to Send?

- Object states:
 - ok, new, mod, meta, del
- Keep a record, for **each** object, and for **each** synchronization partner of:
 - state (above)
 - last-modification time
 - hash
 - metadata



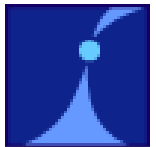
When to Send? II

- Update list on each side, exchange
- Do The Right Thing
 - Both sides decide the same thing
- Handle conflicts! Two rules:
 - Don't lose **anybody's** changes
 - Make it obvious there was a conflict



pphaneuf's Gremlins

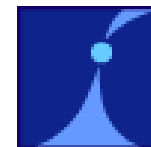
- Deletes
- Renames



Goofball Special Cases

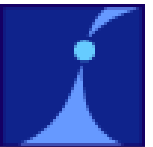


- “Add – Modify – Delete” order...
 - a refers to b.
 - X changes b.
 - Y turns b into a directory. Conflict!
 - Mark b as add, since stuff in it is new! But a is still only modified!
- Worse still, what if a transfer fails after the conflict? What state do you store?



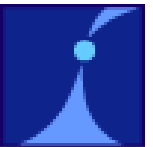
The Grand Scheme

- Step 1: Synchronize files
 - “Offline mode”
- Step 2: “Persistent mode”
 - Use DaZuko or something similar
- Step 3: Boring stuff
 - Security, access control, bleah
- Step 4: Go bananas
 - Object-access API for “online mode”
 - More types of objects



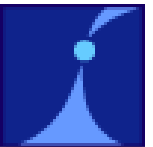
What About Simias / iFolder?

- Confession: I have not read **all** of the Simias code.
- Observations / speculations:
 - Half of every conflict is lost
 - Lack of delta compression
 - Seems to be only for files
 - Already does “persistent mode”, which is impressive!



Demonstration

- ...of something that isn't WvSync!
 - but at least I wrote it.

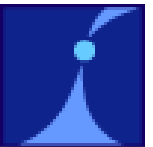


Links

Dave Coombs <dcoombs@nit.ca>

<http://open.nit.ca/wiki/?WvSync>

<http://people.nit.ca/~dcoombs/gvadec/>



Any Questions?

Dave Coombs <dcoombs@nit.ca>

<http://open.nit.ca/wiki/?WvSync>

<http://people.nit.ca/~dcoombs/gvadec/>

