P2P and File Synchronization in Fuego Core

16.12.2005

Tancred Lindholm

Fuego Core Project

Helsinki Institute for Information Technology

http://www.hiit.fi

P2P - Our Understanding

- P2P is a broad term; our understanding is:
  - Provides data or processing distribution
  - Provides redundancy, availability
  - Royal Roads: data availability not dependent on single server
  - Connection topology closer to random net than star
  - Approximately same functionality at each node
  - Sharing of immutable data (usually audiovisual)

Divergence

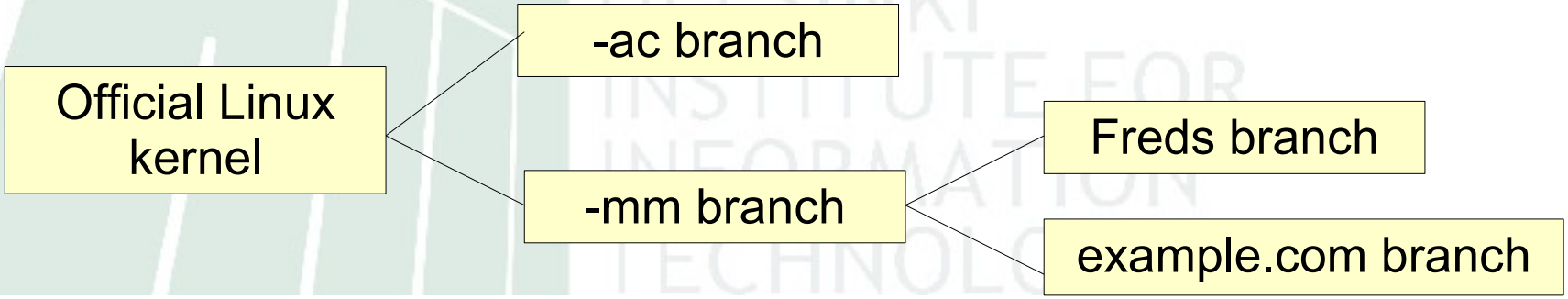
- Mobile environment means intermittent disconnection
- Yet we want always available data → optimistic sharing
- Data may be updated regardless of being disconnected
- This will lead data **divergence**, i.e. multiple variants of an object (e.g. calendar@phone, calendar@workstation)
- How do we deal with divergence?
  - A1: Divergence is an anomaly which is fixed (magically) by the system
  - A2: We need to manage divergence in a way that the user is comfortable with
- Fuego Core picks A2

Divergence and P2P

- P2P best practice is for sharing immutable data, but we have changing data?
- Some faulty reasoning:
  - We have: divergent data at the nodes
  - P2P: replicating same data at nodes for availability
  - → P2P and data divergence do not mix?
- No, P2P is quite useful with divergent data as long as we choose our object identities appropriately
- So how does it fit together (in Fuego Core)?

Fuego Core Sharing Model

- Objects identified by local name + location id, e.g. thesis@laptop, thesis@workstation
- Objects with same or related content are associated to each other with links: thesis@laptop —linkedto→ thesis@workstation
- To synchronize t@laptop, download t@workstation, merge, and upload any changes to t@laptop to t@workstation
- Links organize variants into trees of base and derived work. Just like in the real world :)



So, Where Does P2P Come In?

- The previous slide looks a lot like a tree topology... But:
- Thanks to location scoping, each object has a linear version history; version numbers tell how fresh the data is
- Linked-to objects may be put onto a P2P Network (e.g. thesis@workstation)
- On synchronization, to get *remote changes*, we get the version with highest version number from the P2P network
- If we have *local changes*, we write these directly to the link target host (in client/server fashion)
- In other words, a **hybrid model**
  - P2P is for distributing content
  - Client/Server is for changing content

Pros and Cons of Our Model

- Pros
  - Variants not hidden from user
  - Efficient content distribution over P2P
  - Very simple content updating mechanism
  - Simple to implement
- Cons
  - Centralized data writes requires server
    - but writes less common, switching link to a variant allows groupwork even if main source is unavailable

Conclusion - Syxaw in 1 Slide

- **Syxaw**: Simple mobile file synchronizer
- Optimistic replication; all files always available
- “Link to the file you need”
- Flexible tree-like synchronization topology
- Anticipating large (1-10GB) storage capacity on phones
- Efficient: minimize no of RPCs, batch transfers
- Reconcile by three-way merging
- Use XML-with-references to handle huge XML

HELSINKI
INSTITUTE FOR
INFORMATION
TECHNOLOGY