Cimbiosys: A platform for content-based partial replication

Venugopalan

Thomas L, Douglas,Meg Walraed-Sullivan, Ted Wobber, Catherine C. Marshall, Amin Vahdat

Presented by Darshan N

Requirement

- People use more than one device to store data
- Portable devices have limit on storage and bandwidth.
- Should be able to define its own content-based filtering criteria.
- Automatically share updates with any new other device encountered.

Typical situations

- Alice iPhones gets synchronize with Bob phone only on selected contents (daughters photos).
- Bob watch movies on his i-pod
  - Syncs his player wit *Youtube, The Daily show*
  - Electronic bill board, advertising new TV show
  - Grab commercial content from trusted sources.
  - Bob subscribes to new show.
  - deleting eventually propagated to home media center

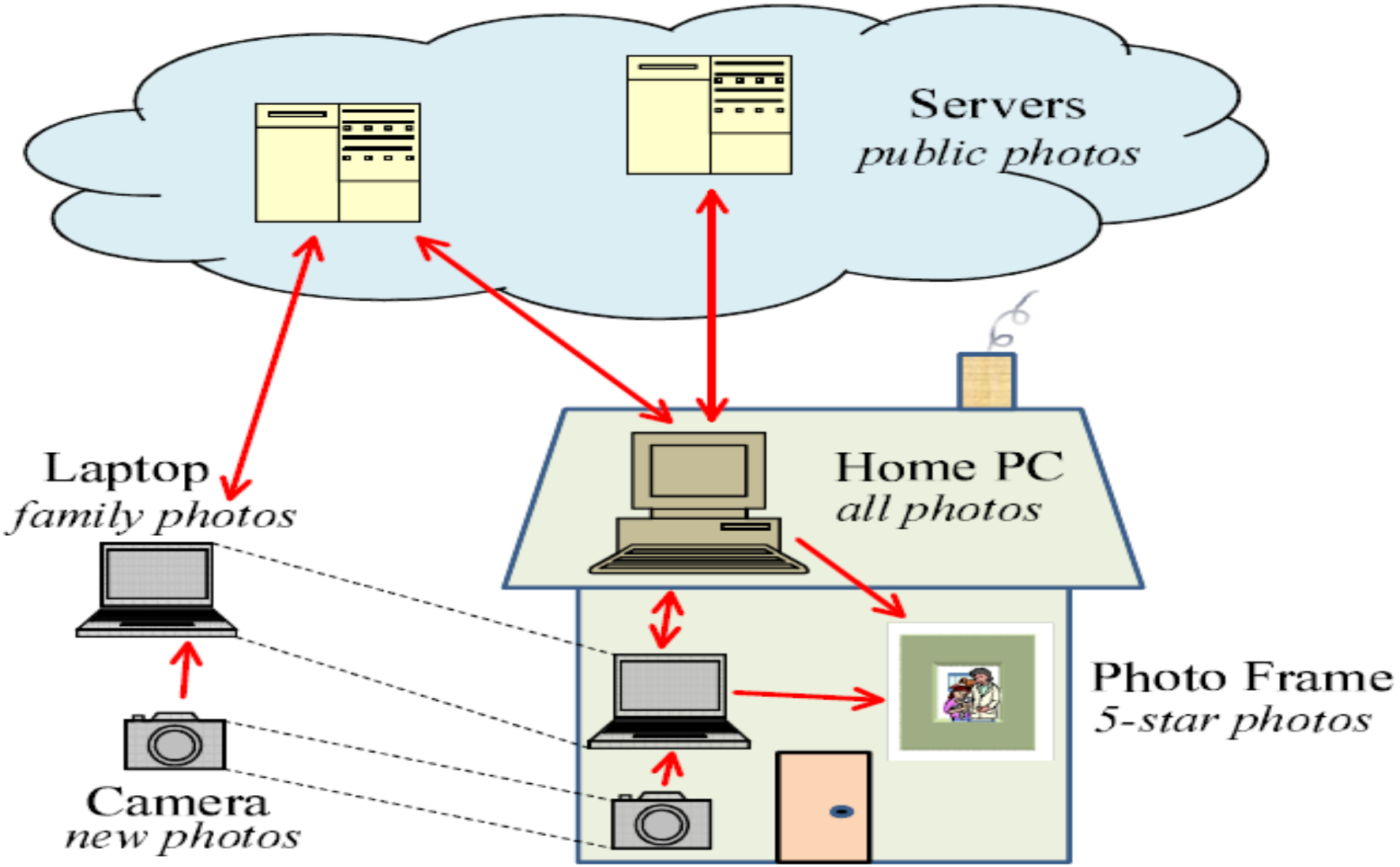

Figure 1: Photo sharing

Challenges for such a activity

- Manage data across their own devices.
- Delivering relevant information to changing set of people and devices
- Requires synchronization through personalized content filtering.
- Updates originate from multiple sites and contributed by different people.
- Inter-device communication may be ad hoc, device proximity, availability of particular content.
- Not all device stores have complete collections

Goals

- It is essential to take advantage of **proximity** and selected replication of content.
- Device represents its **metadata** in a compact form
- State **proportional** to the number of devices and not total number of items.
- Frequent and low BW sync.
- Items latest version meets arbitrary filter criteria are stored.
- independent of any hierarchical namespace.

Outline of the topics

- **Cimbiosys Platform and its feature**
- **Software components**
- **Implementation**
- **CIM Sync protocol**
  - **Knowledge**
  - **Move-out notifications**
  - **Out-of-filter updates**
  - Changing filter
  - **Compaction**
- **Tree Topologies to guarantee synchronization**
- **Evaluation: Simulation and Experiment results**
- **Limitations of the paper**

Cimbiosys Platform

- *Def: Cimbiosys*, an application platform that supports content-based partial replication and synchronization with arbitrary peers, to manage home media and shared data (Replication system), such as calendars, videos etc.
- *Eventual knowledge singularity:*
  - replicas exchange compact metadata , efficient use of BW and resources,
  - Synchronizing , detect overlapping interests, identify missing versions.
  - Knowledge converges to a size,
- *Eventual filter consistency :*
  - over time
  - all items meeting “content-based filtering”

Problems with exisiting solutions

- such as,
  - Ficus, per-item version vectors
  - PRACTI and Bayou, has lesser overhead but, Exchange of operation logs, storage use grow proportion to update rate.
- They are, non optimized BW utilization since, proportional to the collection size or dependent on the update rate, as collection size grows (frequent update)

Features and components

- Provides distributed, peer-to-peer architecture each node is called *Device*.
- Device Stores full or partial copies of one or more data collections.
- *Items* = XML object + associated file (optional)
- *XML object*, eg: photo, created, resolution, quality rating , keyword.
- *Replica*, copy of some or all items in a given collection
- *Filter*, set of item is specified by…
- The default "*" filter indicates, device interested in all items, and hence stores a full replica of the collection.

Cont…

- Device-to-device S*ynchronization* protocol, used to send updated items to other replicas.
- Devices generally have regular partners or any replica they encounter.
- Device join system by creating empty replica.
- Sync protocol to ensure *eventual filter consistency.*

Software components

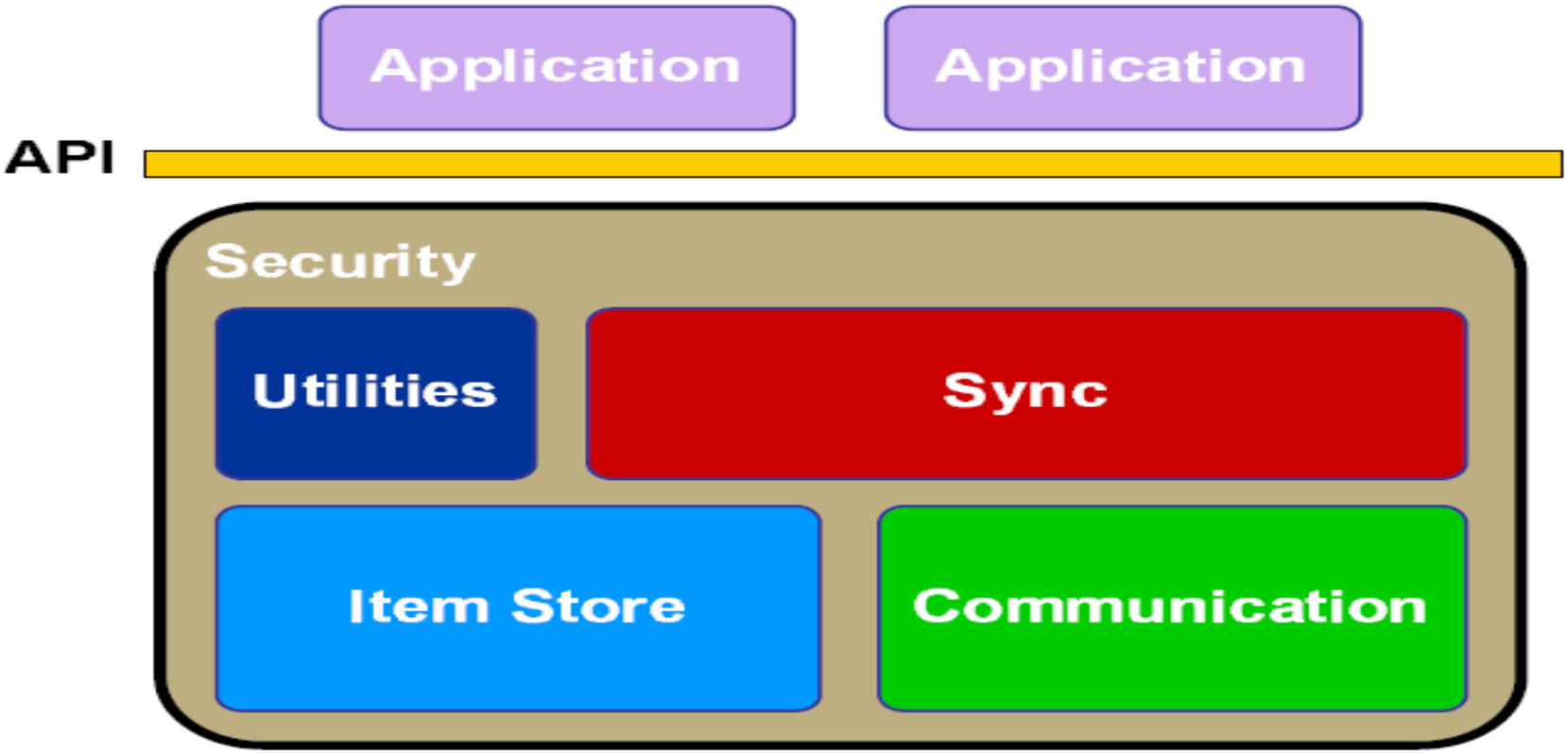

Figure 2: Cimbiosys software architecture

Cont..

- ***Item Store*** *manages* items for local replicas of one or more collections & per item (id, ver-id,xml+file content and information)
- ***Communication***: Both devices agree on network and protocol
  - Network Ethernet, WiFi, cellular, or Bluetooth.
  - Transport protocols such as SOAP-based RPC, HTTP etc,
- ***Sync****:* implements synchronization protocol
- ***Utilities****:* recording synchronization partners, naming collections and devices, managing access controls
- ***Security***: items are digitally signed by the originating device and collection-specific Policies dictate as in which devices can create/update/delete items in a collection
- API's are mean by which application talk to Cimbiosys platform for operation on items/ collections/ local replica , initiate sync etc ..

Implementation

- Two different environments
- C# using Microsoft's .NET and Mace which is a C++ language extension that supports distributed systems development.
- Applications
  - **Cimetric,** collaborative authoring tool writing a paper and
  - **CimBib** is designed as a bibliographic database and personal digital library for people to share references local and remote copies of published papers,

CIM Sync protocol

Metadata

- sync protocol relies on a variety of per-item and per-replica metadata.
- unique identifier for collection and every item
- Each version of an item has unique identifier called *version-id*
- When item is created, updated, or deleted a **new version-id** <replica's id, counter of updates>
- Deleted items marked & referred as *tombstones.*

- The Knowledge is just set of version-ids and consists of identifiers for any versions that

  (a) match the replica's filter and are stored in its item store,

  (b) are known to be obsolete, or

  (c) are known to not match the replica's filter

- Knowledge has one or more fragments (version vector and an associated explicit set of item ids)
- if a replica holds a **" knowledge fragment S:V "** then the replica knows all versions of items in the set S whose version-ids included version vector V .

CIM Sync: Protocol

- one-way, pull-style synchronization protocol.
- *target replica* → *source replica, through* *SyncRequest* message, includes target's knowledge & its filter
- source replica checks - item store for items, whose version-ids unknown to target replica & whose XML contents match the target's filter
- Source responds XML contents, file contents & metadata for each of these items and *SyncComplete* message.
- All messages received by target result in updates is applied to item store.

Example

- Synchronization of photo frame ‘B’ and laptop ‘C’
- State of devices, metadata and item store before synchronization.
- more recent version of *r* and a new item *s*
- updated *k* to reduce its rating

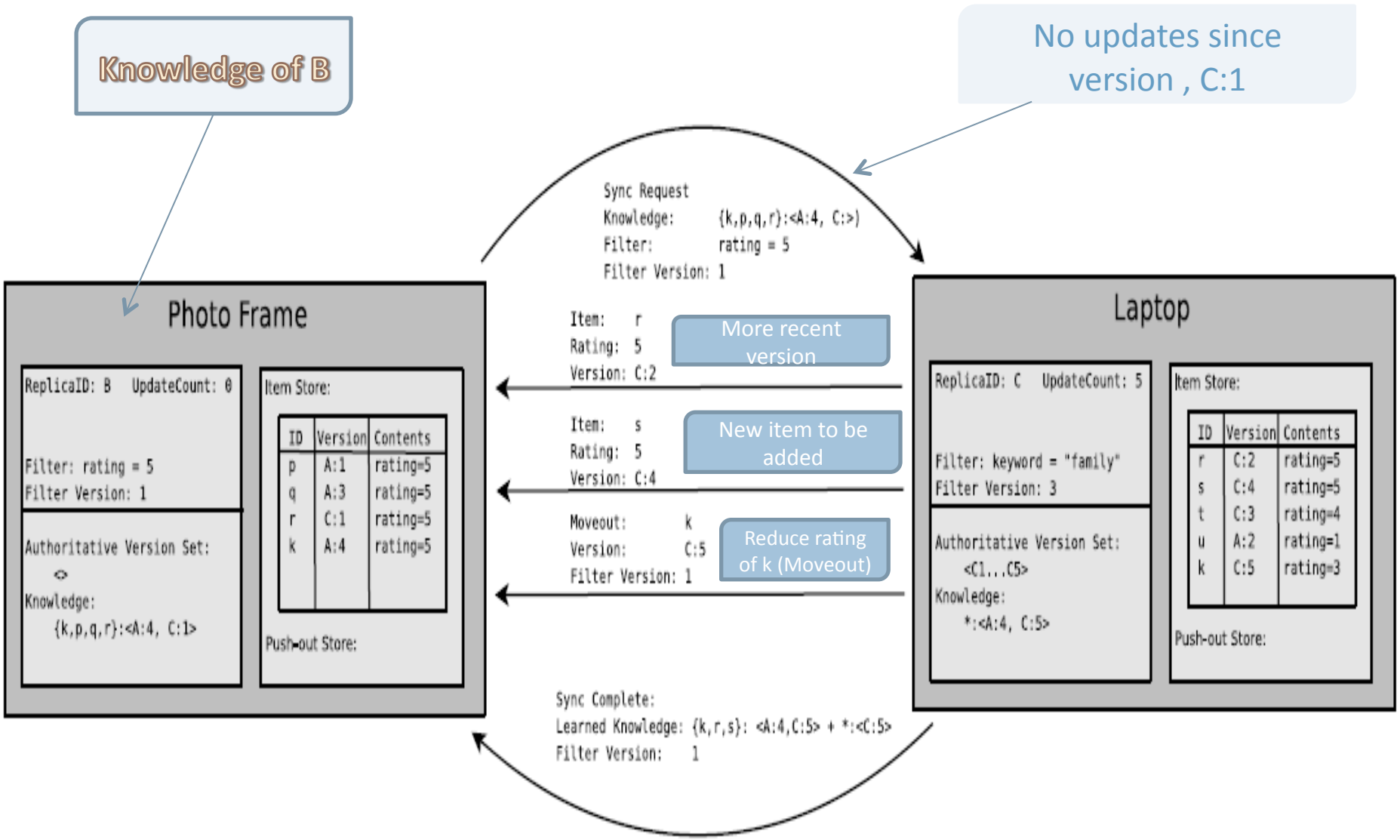

Figure 3: Example synchronization between two replicas

Knowledge

- If source replica's filter no more restrictive than target's filter, i.e item that matches target's filter also matches source's filter, *SyncComplete* message.

- target has broader /incomparable filter
  - Example family and 5*, send only partial knowledge

Move-out notifications

- Source replica tells, causing the target to remove specified items from its item store(later version available).
- *Move-out notifications* during sync.. Will move it out of its *item store* Eg: for *k*
- C1: partial replica currently stored items have later versions that no longer match its filter
- C2: Source has item what target does not have hence move out. To avoid this target tells set of items it stores initially.

Cont…

- Pro: Move-out notifications based on previous filter that no longer applies,(Since target can change its filter anytime)
- Sol: Target Replica increments counter when ever it updates filter & ignore that move-out from source .

Out-of-filter updates

- Operations that produce new versions of items that do not match the local replica's filter are called *out-of-filter updates.*
- Eg: Filter is unread Email and should delete after reading.
- Cannot be discarded immediately
- Items are moved to Special portion, *push-out store,* in visible to applications
- ***Hot potato problem* !!** Send to target (whose filter matches) and delete, but target send it back and delete.

Changing filter

- Case1:Cannot directly remove items of the old filter if it does not match its new filter. Because it may be the only replica making the latest version of such a item.
- Hence push-out store and then eventually deleted.
- Case2: During sync as a source replica. maintain *soft-state* mapping their partners filter versions to the actual filter queries.
- If soft-state discarded, then special error, causing target to resend its full filter as a new sync request.

Knowledge and compaction **

- Replicas items version-ids to their knowledge but not obsolete or deleted items.
- As data exchanged, Knowledge fragments keeps growing, hence compaction.
  - S1 is a subset of set S2 and the version vector V2 dominates V1 (any V1 is also present inV2)
  - S1 and S2 can be combined into a single knowledge fragment
- Cannot be done always
  - keyword k, s , t and C:1 become C :2, C:3,C:4
  - which are all C filter different and will not be updated

Compaction rules for any pair of knowledge fragments

$S_1{:}V_1 + S_2{:}V_2 \Rightarrow$

	$S_1 \subset S_2$	$S_1 = S_2$	$S_1 \supset S_2$	*otherwise*
$V_1 \subset V_2$	$S_2{:}V_2$	$S_2{:}V_2$	$S_2{:}V_2$ + $S_1\text{-}S_2{:}V_1$	$S_2{:}V_2$ + $S_1\text{-}S_2{:}V_1$
$V_1 = V_2$	$S_2{:}V_2$	$S_1{:}V_1$	$S_1{:}V_1$	$S_1 \cup S_2{:}V_1$
$V_1 \supset V_2$	$S_1{:}V_1$ + $S_2\text{-}S_1{:}V_2$	$S_1{:}V_1$	$S_1{:}V_1$	$S_1{:}V_1$ + $S_2\text{-}S_1{:}V_2$
otherwise	$S_1{:}V_1 \cup V_2$ + $S_2\text{-}S_1{:}V_2$	$S_1{:}V_1 \cup V_2$	$S_1{:}V_1 \cup V_2$ + $S_1\text{-}S_2{:}V_1$	$S_1{:}V_1$ + $S_2{:}V_2$

Figure 4: Knowledge compaction rules

Tree Topologies to guarantee synchronization

- Forces replica, replicas of a given collection to configure themselves into a hierarchically filtered tree topology.
- Synchronization topology, is to achieve desired system properties, eventual knowledge singularity.
- Root /Reference replica * filter
- Creating New replica at least root will be because its * least restrictive.
- Retire graceful from a collection, should notify its children so they can select a new parent
- Immediate parent or one of the child become parents for other siblings
- Replica can changing the parent: but suitable filter

Benefits of TST

1. Well-connected and groups of replicas for the same collection cannot be disconnected indefinitely, hence eventual convergence.
2. Each version of an item has a guaranteed path, Eg: new version created flows up the tree from child to parent replicas until it reaches common ancestors.
3. Move-out notifications can be delivered by a parent to any of its children, source replica **no more** restrictive than the target.
4. Push-out store flow up tree, until they reach replicas interested with the item.
   - The tree topology prevents replicas from playing “hot potato” with out-of-filter versions.
5. Ensures eventual knowledge singularity: As authoritative versions are passed up the tree, a parent replica assumes authority for any versions generated by any of its children or descendants.
   - all authoritative versions arrive at the reference replica, which then produces a single star-knowledge

Simulation

- Three serial phases: with 10 replicas (3 level hierarchy)

1. 1000 inserts of new items at randomly chosen replicas,
2. 400 syncs between randomly chosen sync partners
3. Another 400 random syncs interleaved with 1000 updates to random items.

- PIVV : maintains individual version vector for each item that matches replicas filter and sends entire knowledge during syncs and receives only those VV pertaining to items in its filter
- CIM-PIVV modified CIM, discards whenever the star knowledge subsumes them.

- CIM and CIM-PIVV, knowledge is fragmente in initial stages
- Significant savings initially whereas savings is marginal synchronizin the updates in the third phase.
- The key reason for this behavior is initial inserts were performed in batches, providing opportunities for CIM to compact knowledge into item sets.

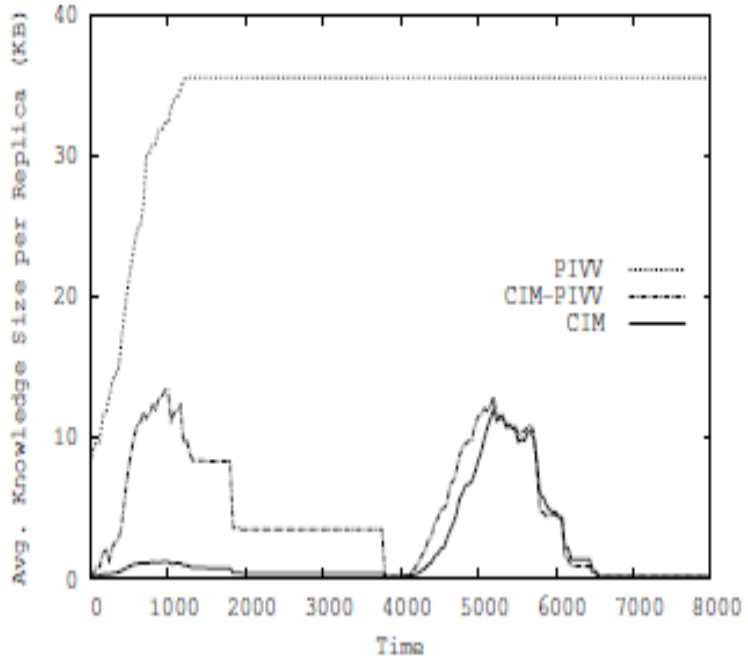

Figure 5: Average size of knowledge per replica vs. time

- All three simulated solutions for partial replication eventually converge (at the same rate)
- Figures 5 and 6 indicates that the system could achieve filter consistency much earlier than knowledge singularity. (as it is across the whole system till '*')

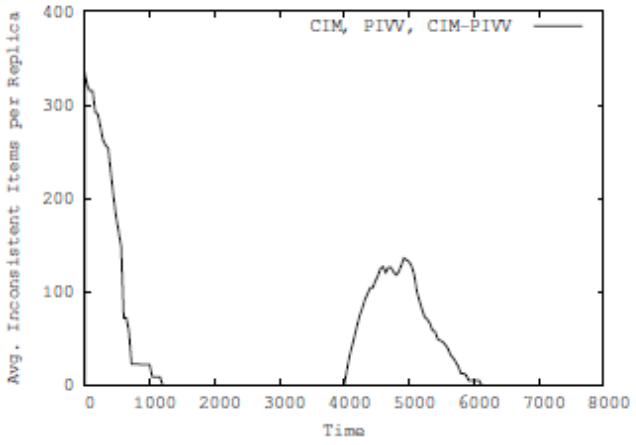

Figure 6: Average inconsistent items per replica vs. time

Experimental Evaluation of Mace implementation

- Bandwidth is measured as the sum of all data as it is queued into the transport layer.
- String of 6 character as data content shouldn't contribute significantly to bandwidth used.

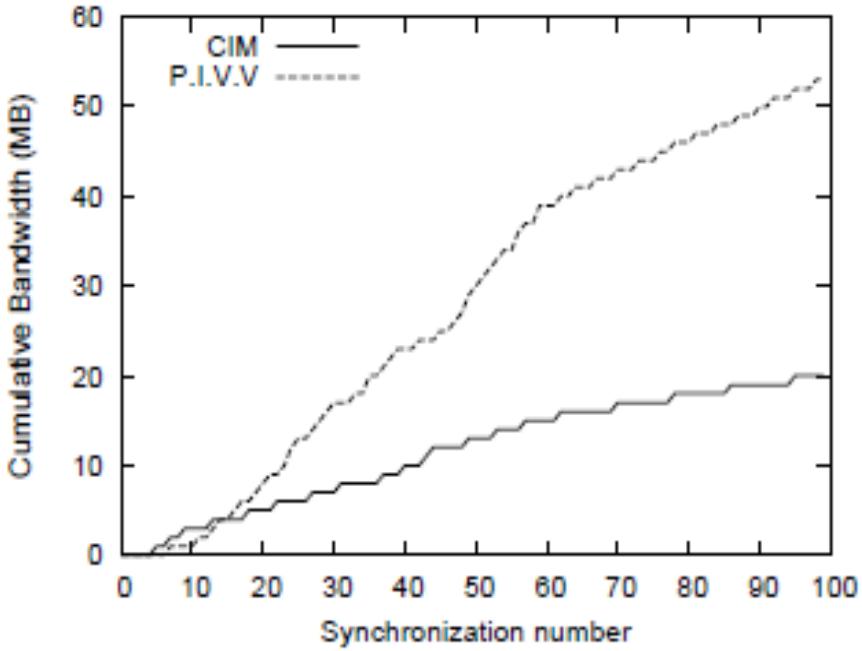

Figure 9: Bandwidth Usage: Reference Replica

- CIM: Spikes demonstrate the significance of the knowledge compaction process. at the completion of the synchronization, the target compacts its knowledge
- PIVV: requires version vector be stored per item, knowledge stored for this representation remains on the order of the number of items in the system.

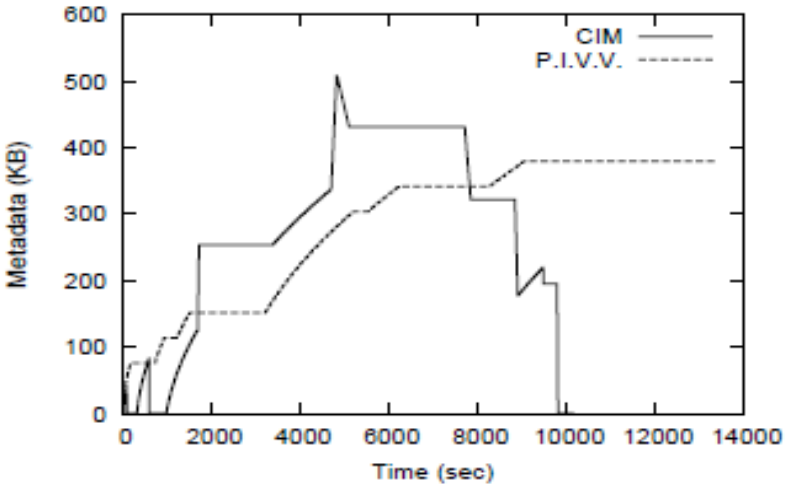

Figure 8: Knowledge Size: Reference Replica

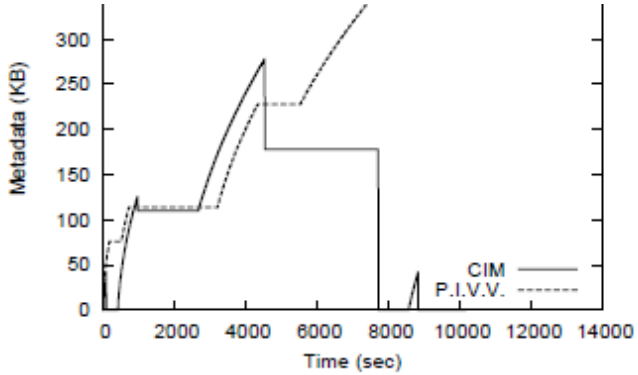

Figure 7: Knowledge Size: Partial Replica

Limitations

- These systems support peer-to-peer sharing of data and can be highly available, but are unable to guarantee correctness in the presence of malfunctioning or malicious nodes.
- Simultaneous updates from more than one replica.
- The restriction that, overall synchronization topology must include an embedded tree with a reference replica.(* knowledge or root of the tree)

Thank you!

QUESTIONS?