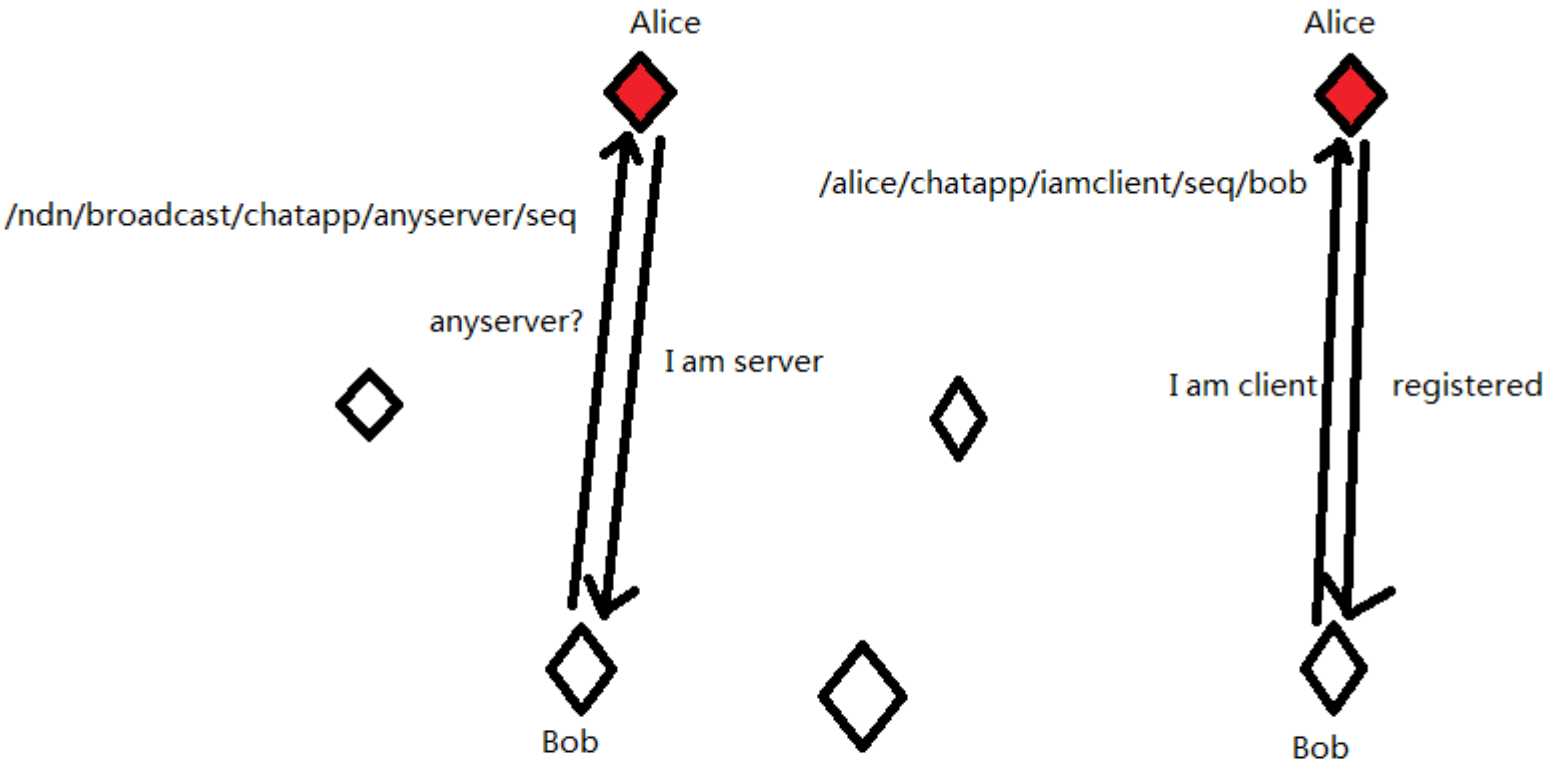
DRSTL Weekly Progress Report II

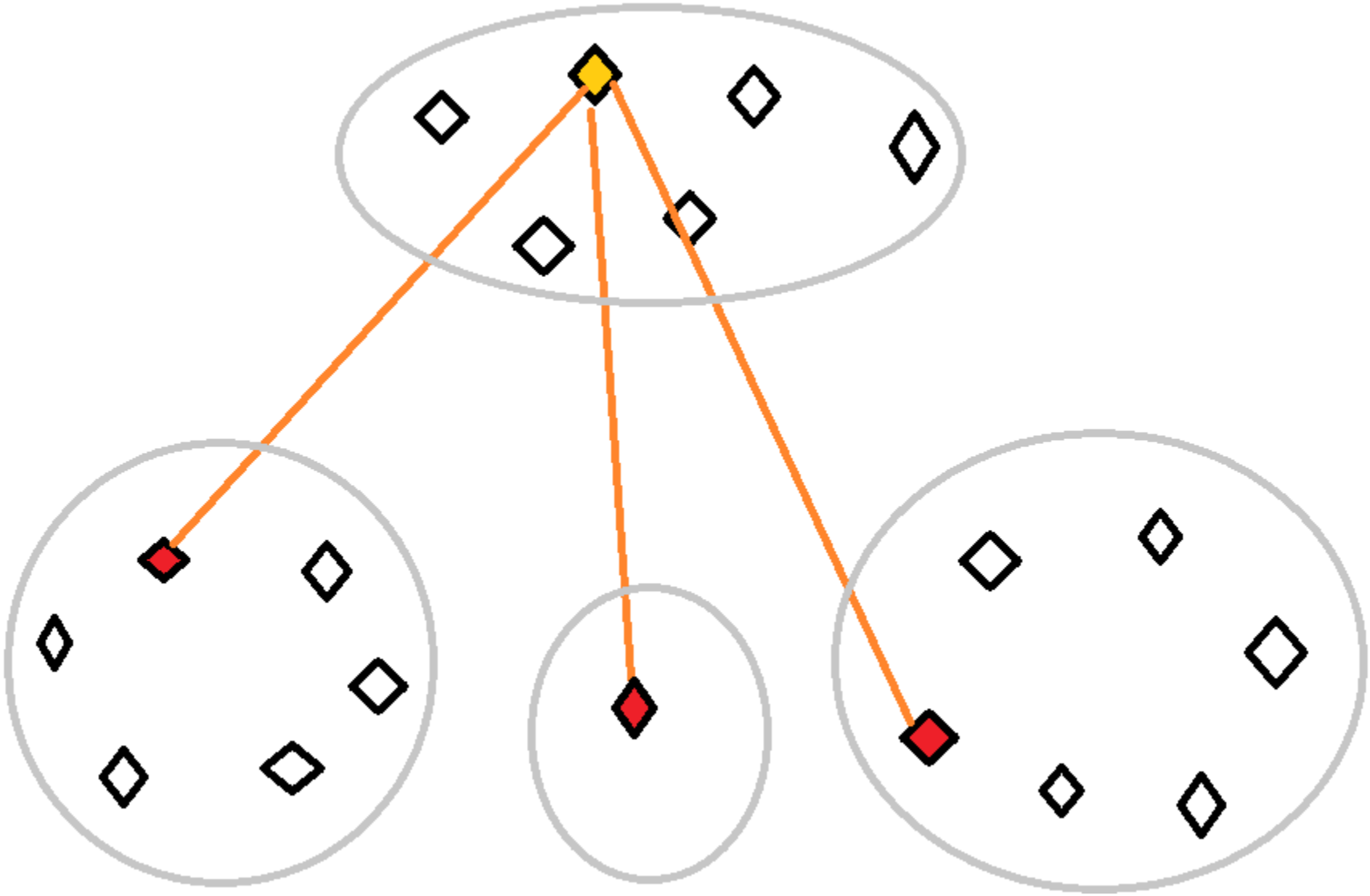
Hebi Li
12/02/2013

Table of Contents

1. Server generation(Review)



1. Server Generation(review)



2. time-label 基于时间的状态

- 每次同步的状态被打上时间的标记
- 总是上一级做的标记，因为 server 主导同步
- 如果包是从上一级来的，则将原来的标记不动，并加上自己的标记，传往下一级
- 标记的内容是自己当前的时间 + 自己的名字 + 自己的 level

2. 基于时间的状态同步过程

- 每个节点向他的 server 发 sync interest ， 这个 interest 的名称如下：

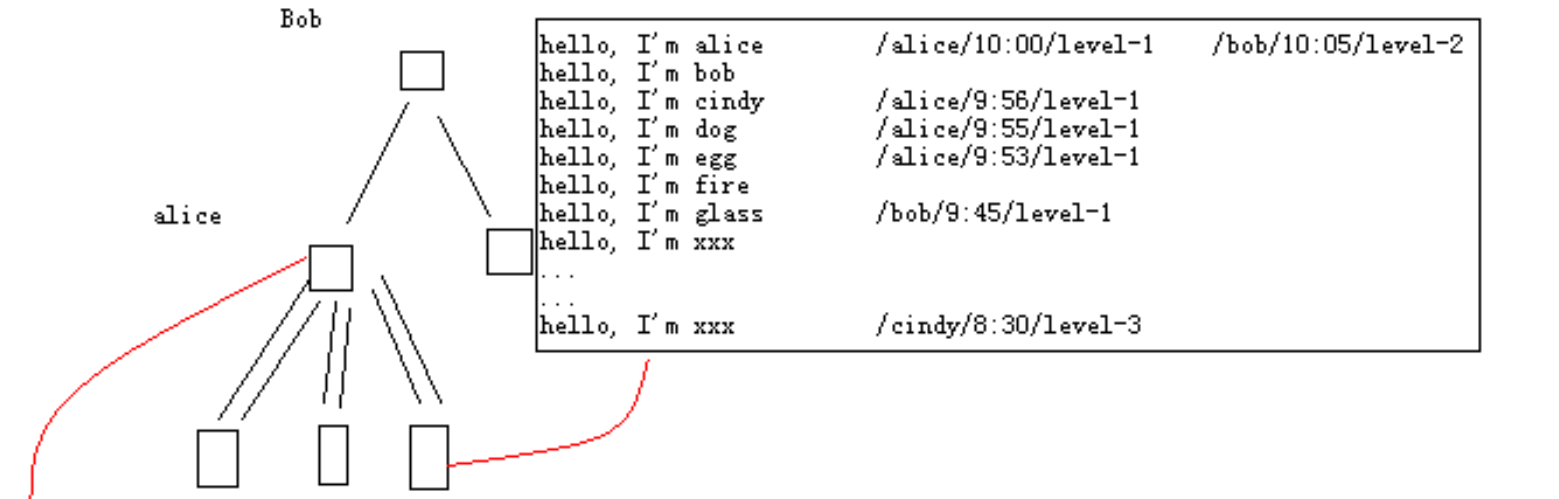
  /alice/chatapp/time-label

  * lice 是此节点的服务器
  * Time-label 是此节点最新的标签，包含其每个上层服务器的最新时间戳。

2. 基于时间的状态同步过程

- Server 收到 sync interest 后，从最低 level 开始，分别对比他的和自己的 time-label （包含之上所有层级的 time-label），若匹配上，则返回此次同步之后的消息

example

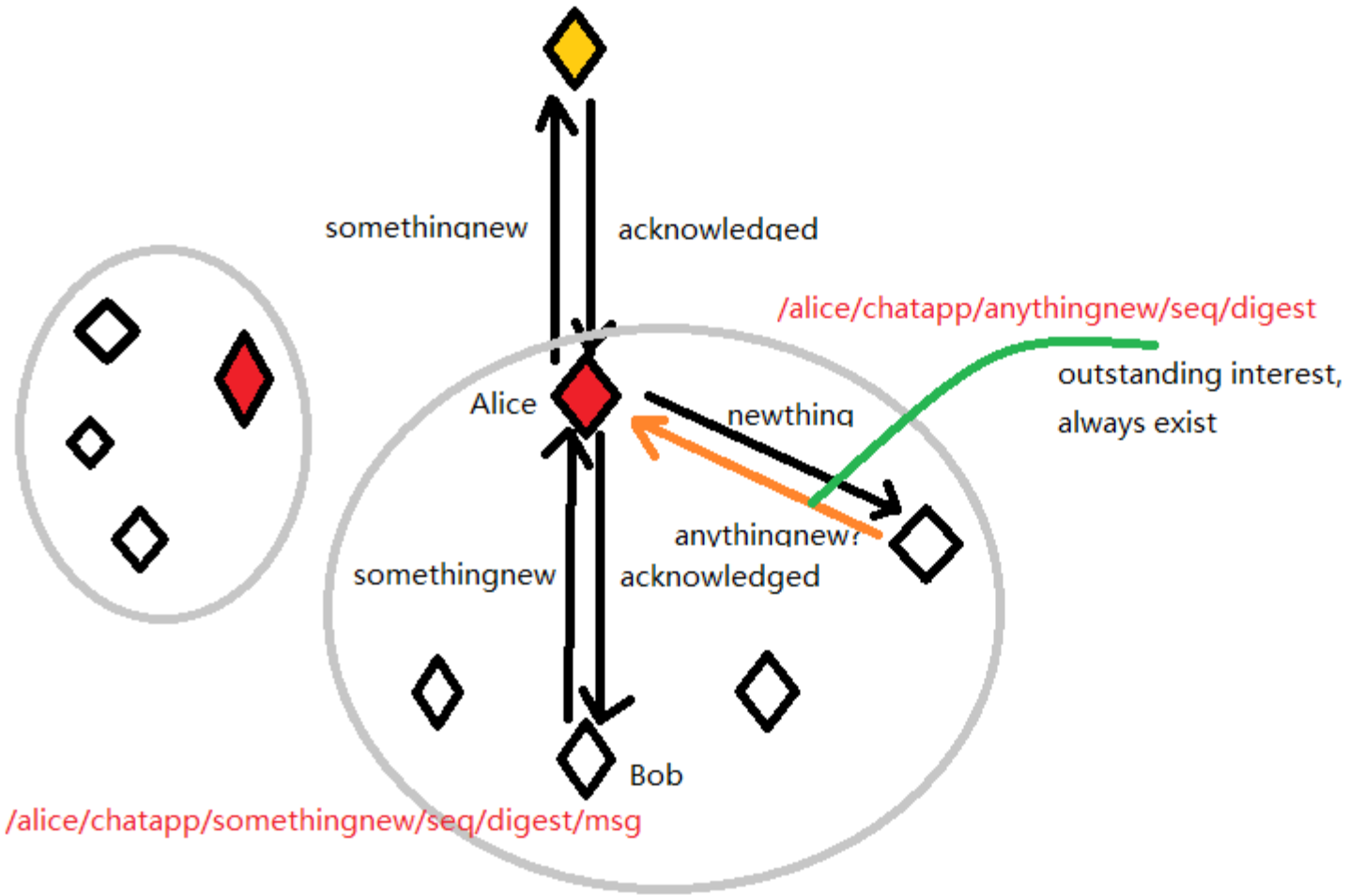


sync interest

/alice/chatapp/alice_10:00_level-1&bob_10:05_level2&cindy_8:30_level-3

```
hello, I'm aaa          /alice/10:07  /bob/10:08
hello, I'm bbb
hello, I'm ccc
hello, I'm alice        /alice/10:00
....
...
...
```

example


somethingnew
acknowledged
/alice/chatapp/anythingnew/seq/digest
outstanding interest,
always exist
Alice
newthing
anythingnew?
somethingnew
acknowledged
Bob
/alice/chatapp/somethingnew/seq/digest/msg

3. Trouble shooting

- 3.1 simutaneous data generation(two sceneries)
- 3.2 server fail
- 3.3 node transfer
- 3.4 root node fail

3.1 simutaneous data generation

- 第一种情况： 同一个 server 的两个 client 同时发消息。
- 解决方法： server 按照 interest 到来的先后顺序排列消息以及加 time-label, 并主导同步。
- 第二种情况：一般的，不同层级上的且只共享非常上层 server 的 clients
- 解决方法：每一层 server 只会根据他自己收到消息的序列排序并加 time-label （这个消息状态可能有上层 time-label ），所以， nothing special.

3.2 Server fail

- Server 下线了
- 解决方法：
  - 1. 识别 server 下线： 每个人往 server 发 heatbeat interest ， server 回这个 interest 。
  - 2. 重新选 server ： 检测到 server 不在了，按照选 server 的方法选出新的 server （每个节点可能归入邻近的 server ，也可能这个区域分裂成多个小区域）
  - 选出的 server 会将接下来的同步信息打上自己的标签。当然这时候组内对于更上层的所有 server 的标签都不受影响，可以互相认识。

3.3 node transfer

- 考虑最复杂的情况： level2 的节点跑到别的分支 level0 上。
- 解决方法：此时他还有所有其上层的同步状态 time-label 。只要原来所在分支和新的分支有交集（最坏的情况交集是顶层 server)，都可以被识别。

3.4 root node fail

- 根节点挂了，则照常选出新的根。此时虽然原来的根不存在了，但是大家还是可以识别原来根的 time-label ，因为每个人还有这些信息。
- 之后的同步包都使用新的根的 time-label 。

- 补充：如果真的网络足够大，为了性能考虑，可能加一些公共服务器会更稳定，但不是必须。比如可以固定顶层或者每个省都固定一个次顶层服务器。

4. Comparation and merits

- 4.1 无需计算 digest
- 4.2 无需保存 digest log
- 4.3 ChronoSync 处理 simultaneous data 时非常麻烦费时。此方法则完全无问题。
- 4.4 ChronoSync 处理长时间下线，根本无法处理。此方法可以有效处理，只需要比对时间，且可以尽可能减少 overhead.
- 4.5 每层 server 相当于收集它的 group 里的信息，然后统一发往上层。所以当发消息很密集的时候， server 可以起到汇集作用，将好多消息转成一个包往上传。
- 4.5 层相对独立，只完成自己的同步以及上层的借口。鲁棒性好。

5. Implement

- 打算用 NS3