BF-based chunk availability compression for PPSP-02

Lingli Deng: denglingli@chinamobile.com
Jin Peng: pengjin@chinamobile.com
Yunfei Zhang: hishigh@gmail.com

IETF 87@Berlin

Outline

- Motivation
- Background
- Analysis
- Proposal
- Discussion
- Example

Motivation

- There are frequent bitmap exchanges in PPSP.
  - Uncompressed bitmap is relatively big (of several KBs).
  - They are exchanged frequently (less than several seconds).
  - It sets a limit to the system's efficiency and scalability
- There are efficiency requirements in PPSP PS
  - PPSP.TP.REQ-3: The tracker protocol MUST take the frequency of messages and efficient use of bandwidth into consideration, when communicating chunk availability information.
  - PPSP.PP.REQ-7: The peer protocol MUST take the frequency of messages and efficient use of bandwidth into consideration, when communicating chunk information.

Background

- Original bitmap Scheme uses a bit-array to represent the chunk set, and mark those bits corresponding to the locally available chunks.
- Con: Its length grows as the number of chunks (n) increases.
- Chunk range scheme uses a array of starting+ending chunk index pairs to represent continuous intervals.
- Byte range scheme uses an array of starting+ending byte index pairs to represent continuous intervals.
- Bin number scheme uses a universally assigned bin number (integer) for any given continuous interval.

Scheme type	Orig-bitmap	Chunk range	Byte-range	Bin number
Bit length	n	2klogn	2kmlogn	klogn

Variables: n - # of chunks; k - # of intervals; m - # of bytes per chunk.

A Further Look into Processing/Storage

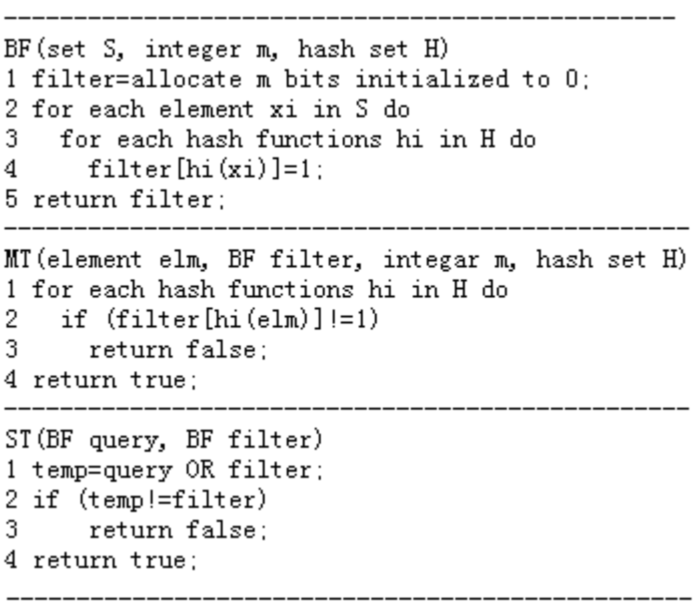
Scheme type	Orig-bitmap	Chunk range	Byte-range	Bin number	???
Bit length	n	2klogn	2kmlogn	klogn	1
Formation	n	n	n	nlogn	1
Single Inquiry	1	2klogn	2kmlogn	logn	1
Group Inquiry(g)	g	2klogn	2kmlogn	glogn	1
Partial Update(p)	p	2klogn	2kmlogn	Tlogn	1
Delta bit len(d)	n	2d	2d	dlogn	1

Variables: n - # of chunks; k - # of intervals; m - # of bytes per chunk.

g - # of chunks in a compacted inquiry;

p/d - # of chunks in a partial chunk bitmap update.

Proposal: BF compression Scheme

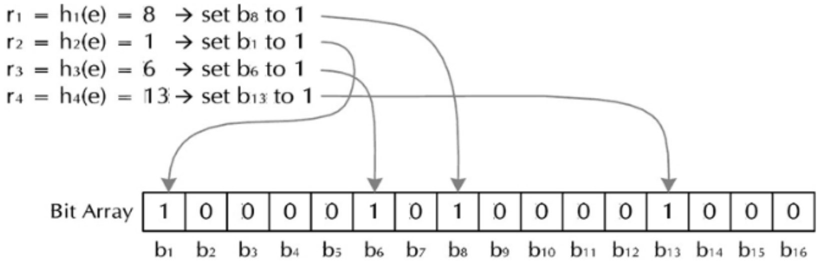


```
BF(set S, integer m, hash set H)
filter=allocate m bits initialized to 0;
for each element xi in S do
  for each hash functions hi in H do
    filter[hi(xi)]=1;
return filter;
```

```
MT(element elm, BF filter, integar m, hash set H)
for each hash functions hi in H do
  if (filter[hi(elm)]!=1)
    return false;
return true;
```

```
ST(BF query, BF filter)
temp=query OR filter;
if (temp!=filter)
    return false;
return true;
```

Figure 1 Basic algorithms for BF-bitmaps.



$r_1 = h_1(e) = 8$ → set b_8 to 1
$r_2 = h_2(e) = 1$ → set b_1 to 1
$r_3 = h_3(e) = 6$ → set b_6 to 1
$r_4 = h_4(e) = 13$ → set b_{13} to 1
Bit Array
1 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0
b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15} b_{16}

Figure 2 Bloom Filters: an example.

High Efficiency

Storage/transmission: Bit length: constant.

Processing: Formation/Inquiry/Update: constant.

Endurable Lose of accuracy

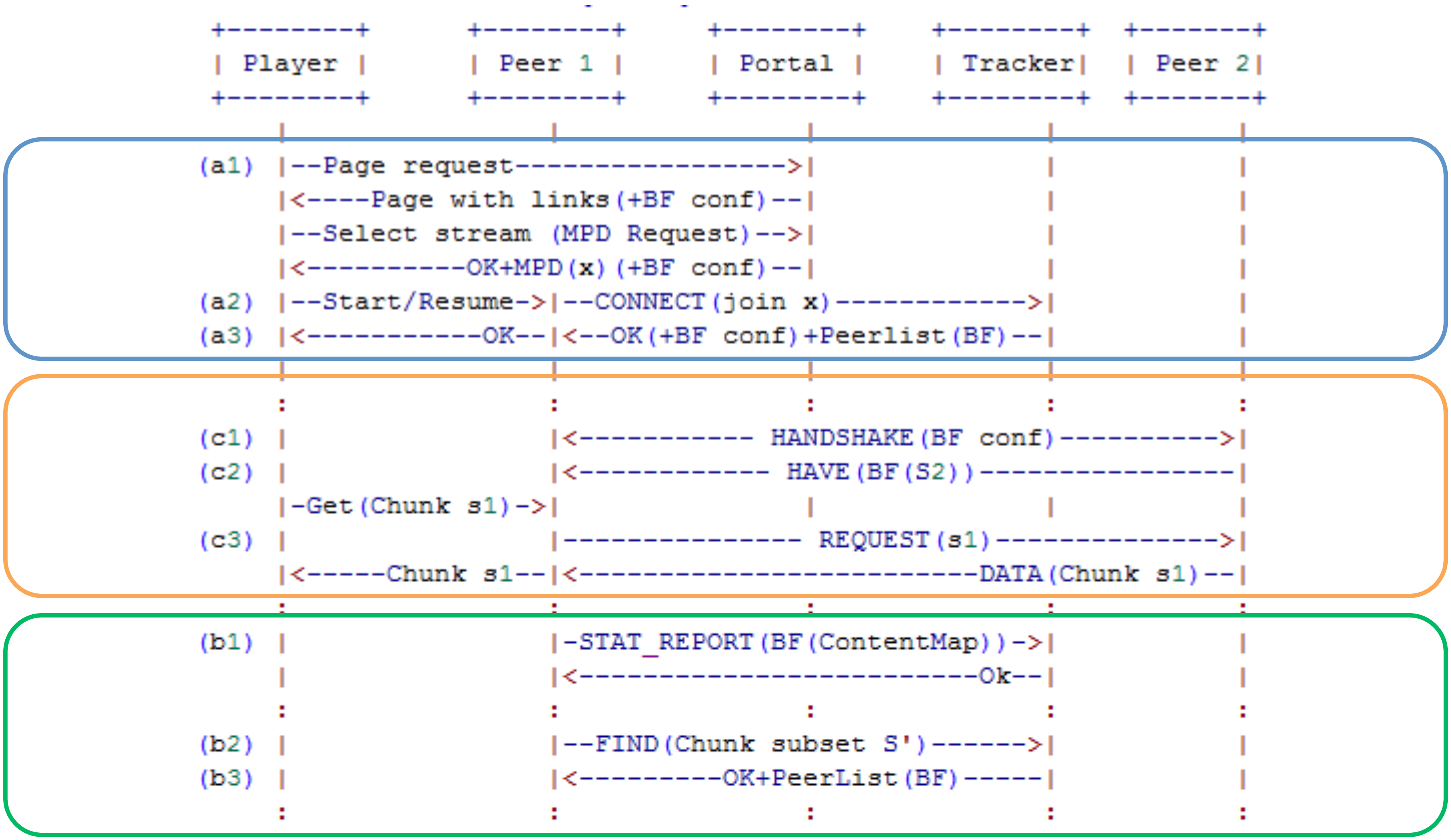
Be controlled by the system configuration of the bit array's length, choice and # of hash functions.

Example: a 2GB movie file, divided into 2MB chunks, whose a 1024-bit original chunk bitmap, can be represented by a 128-bit BF-bitmap (using 4 hashes), with only 3% mis-hits rate.

Suggestions for Discussion

- RECOMMENDED for PPSP-TP-base/extended
  - Strictly controllable cost for a central entity
    - constant bit-length irrelevant of the chunk-set
    - only replacement or simple bitwise operations needed
  - Certain mis-hits rate COULD be tolerable
    - Tracker serves as an initial broker for neighboring peers
- OPTIONAL for PPSPP
  - Peers willing to trade accuracy with cost-efficiency
    - Peers with limited computation/memory resources
    - Peers with huge number of concurrent links, e.g. SNs
  - Certain mis-hits rate MAY be tolerable
    - REQUEST and DATA SHOULD use the original chunk id.

Example: PPSP with BF-bitmaps



```
     +---------+      +---------+    +---------+   +---------+ +--------+
     | Player  |      | Peer 1  |    | Portal  |   | Tracker| | Peer 2|
     +---------+      +---------+    +---------+   +---------+ +--------+
          |                |               |              |           |
     (a1) |--Page request----------------->|              |           |
          |<----Page with links(+BF conf)--|              |           |
          |--Select stream (MPD Request)-->|              |           |
          |<----------OK+MPD(x)(+BF conf)--|              |           |
     (a2) |--Start/Resume->|--CONNECT(join x)------------>|           |
     (a3) |<-----------OK--|<--OK(+BF conf)+Peerlist(BF)--|           |
          |                |               |              |           |
          :                :               :              :           :
     (c1) |                |<----------- HANDSHAKE(BF conf)---------->|
     (c2) |                |<------------ HAVE(BF(S2))----------------|
          |-Get(Chunk s1)->|               |              |           |
     (c3) |                |--------------- REQUEST(s1)-------------->|
          |<-----Chunk s1--|<-------------------------DATA(Chunk s1)--|
          :                :               :              :           :
     (b1) |                |-STAT_REPORT(BF(ContentMap))->|           |
          |                |<--------------------------Ok--|           |
          :                :               :              :           :
     (b2) |                |--FIND(Chunk subset S')------>|           |
     (b3) |                |<---------OK+PeerList(BF)-----|           |
          :                :               :              :           :
```

THANK YOU!