PPSP
Internet-Draft
Intended status: Standards Track
Expires: August 16, 2014

L. Deng
J. Peng
China Mobile
Y. Zhang
CoolPad
Y. Huang
Huawei

February 12, 2014

Efficient Chunk Availability Compression for PPSP
draft-deng-ppsp-bfbitmap-05.txt

Abstract

Bloom filters are proposed to be used in compressing chunk availability information, periodically exchanged between peers and the tracker through PPSP-TP and PPSPP protocols, to reduce relevant cost and enhance scalability.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 16, 2014.

Copyright Notice

Table of Contents

1. Introduction

As it is pointed out by [I-D.ietf-ppsp-problem-statement], current P2P streaming practices often use a "bitmap" message to exchange chunk availability. The message is of kilobytes in size and exchanged frequently, e.g., an interval of several seconds or less.

To begin with, in a mobile environment with scarce bandwidth, the message size may need to be shortened or it may require more efficient methods for expressing and distributing chunk availability information, which is different from wire-line P2P streaming.

Even in a wire-line P2P streaming application, frequent exchange of large volume of bitmap information, is among the key factors that set a limit to the system's efficiency and scalability [P2P-limit].

Therefore, the following requirements for PPSP protocols in terms of chunk availability exchange are stated in [I-D.ietf-ppsp-problem-statement] :

> 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.

In this draft, we propose an efficient bitmap compression scheme for chunk availability information in PPSP protocols. Given the Bloom Filters' wide applications in Internet and demonstrated efficiency with highly compacted data structure and low complexity and cost in terms of information storage, transportation and computation, it is expected to relieve a PPSP implementer from the dilemma between "the frequency of messages" (i.e. the timely exchange of information that contributes to better user experience) and "efficient use of bandwidth" (i.e. the limit of a single node/peer that holds the system's overall scalability by throat).

2. Background on Bloom Filter

Bloom Filter (or BF for short) was first introduced in 1970s [BF-bloom], which makes use of multiple hashing functions to build a mapping from a set of elements to a compact binary array, to realize highly efficient member queries with a tolerably low error rate of wrongly reported hits. Despite their extraordinary efficiency in terms of storage reduction and query acceleration, BFs suffer from the fact that there is possibility that a non-member of the set be wrongly taken as a member after the query. However, research [BF-analysis] shows that the odds that a BF-based membership query makes an erroneous hit can be suppressed to near zero, by a tactful configuration of various system parameters, including the hash functions used, the number of hash functions to be used, and the length of the bit array.

```
-------------------------------------------------
BF(set S, integer m, hash set H)
1 filter=allocate m bits initialized to 0;
2 for each element xi in S do
3   for each hash functions hi in H do
4     filter[hi(xi)]=1;
5 return filter;
-------------------------------------------------
MT(element elm, BF filter, integer m, hash set H)
1 for each hash functions hi in H do
2   if (filter[hi(elm)]!=1)
3     return false;
4 return true;
-------------------------------------------------
ST(BF query, BF filter)
1 temp=query OR filter;
2 if (temp!=filter)
3     return false;
4 return true;
-------------------------------------------------
```

Figure 1: Basic algorithms for BF-bitmap

As shown in Figure 1, the BF(S,m) algorithm takes a n-membered subset S={x1,x2,...,xn} from a universal set U as input, and outputs a m-bit binary array B as a compacted representation of S. In order to do that, it makes use of k independent random hash functions, each of which maps a member to a marked bit in B (i.e hj: U-> [1,m], j=1...k). The BF algorithm is highly efficient in the following aspects:

- It is quite simple and straightforward to generate the BF representation of a set S, B=BF(S): initially, all the bits in B is set to 0; then, for each member x of the set S, mark each bit in B, to which a hash function maps x (as shown in Figure 1 as the BF algorithm).
- It is highly efficient to check whether or not a given element x is in any BF-represented set B=BF(S): for each hash function hj, check the value of B[hj(x)] against 1. It is always safe to exclude the element x out of set S, once there is a zero-valued hash bit. Otherwise it is assumed that x is a member of S (the MT algorithm in Figure 1).
- It is also highly efficient to check whether or not a given element set S is contained by another set S' if they are both represented as BF-bitmaps, say query(=BF(S)) and filter(=BF(S')) for instance. It is always safe to return "false" to the requestor if there's any marked bits in BF(S') and not marked in

BF(S), which can be realized by two simple bitwise operations(as shown by the ST algorithm in Figure 1).

For instance, given a 2GB movie file, the original bitmap for a sharing peer would be 1024-bit (if the system is using 2MB-sized segments). By simply using 4 uniform random hash functions and a 128-bit BF-bitmap, the possibility of erroneous hits by MT algorithm would be lower than 3%.

As for a simple illustration, the 4 hash functions may be established through the MD5 message-digest algorithm [RFC1321], a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value from an arbitrary binary input. MD5 has been utilized in a wide variety of security applications, and is also commonly used to check data integrity.

Specifically, the processing of 4 hash functions is as follows: use the MD5 algorithm to turn a given chunk_ID into a 128-bit binary array, further separate the 128 bits into 4 arrays (32-bit each), and finally divide each of them using 128 to yield 4 integers in the range of [1,m].

3. BF-based Chunk Availability Exchange

We first construct a general message flow (shown in Figure 2) from PPSP protocols, and then discuss how to integrate BF-bitmap algorithm with it.

3.1. A non-BF PPSP session

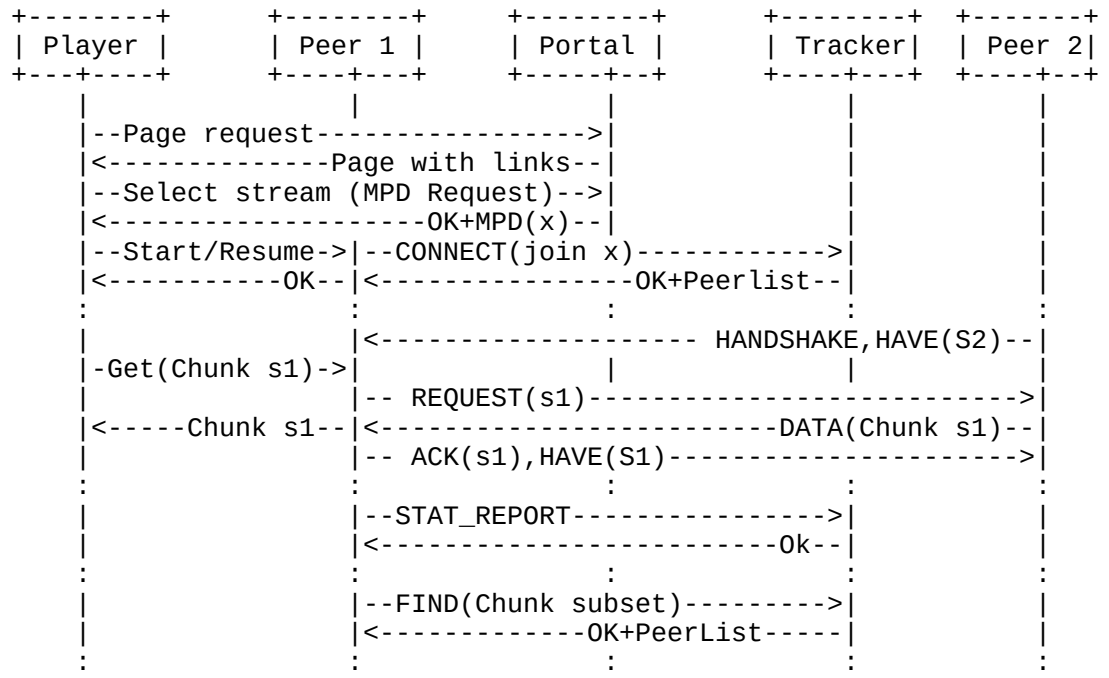

Figure 2: A typical PPSP session for watching a streaming content.

When a peer wants to receive streaming of a selected content (Leech mode):

1. Peer connects to a connection tracker (which may be located through a web portal) and joins a swarm.
2. Peer acquires a list of other peers in the swarm from the connection tracker (via the tracker protocol) through the CONNECT message.
3. Peer exchanges its content availability with the peers on the obtained peer list (via peer protocol) through the HAVE message.
4. Peer requests content from the identified peers (via peer protocol) through the REQUEST-DATA messages.
5. Peer periodically reports its status and chunk availability with the tracker (via the tracker protocol) through the STAT_REPORT message.
6. Peer acquires a list of other peers for a specific subset of media chunks in the swarm from the connection tracer (via the tracker protocol) through the FIND message.

3.2. A PPSP Session with BF-bitmaps

This document proposes to employ bloom filter algorithms in compressing chunk availability information exchanged and stored between peers and the tracker through the PPSP-TP protocols and PPSPP protocol. Relevant extensions to the current protocols are summarized as follows: (as shown in Figure 3)

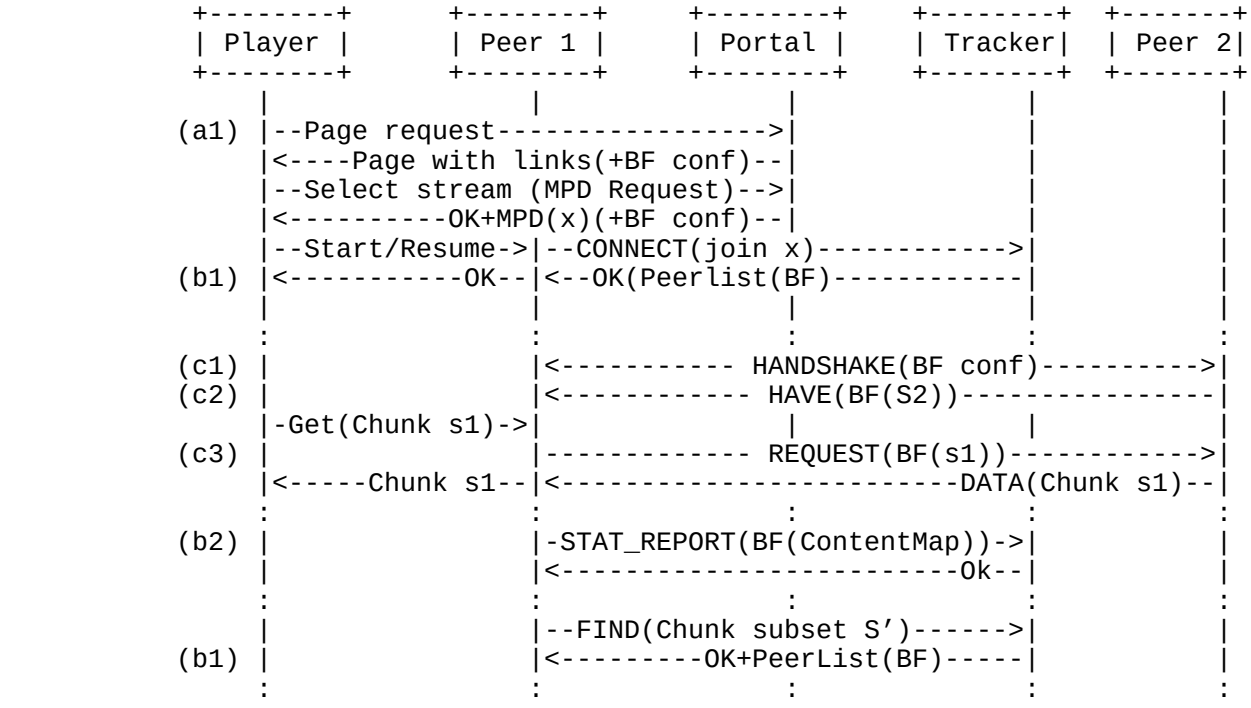

Figure 3: A typical PPSP session with BF-bitmaps.

a. Configuration Setup:

* (a1)Configuration Setup: m, The length of the output bit array and H, the hash functions in use, are system level parameters that should be configured globally: (a1) the BF configurations (or BF conf for short) be stored at the web portal and published to a requesting peer through the web page or MPD file transaction, which could be incorporated to the "Installation and Initial Setup" in PPSP TP protocol [I-D.ietf-ppsp-base-tracker-protocol].

b. Integration to the extended TP protocol [I-D.ietf-huang-extended-tracker-protocol]: In the extended TP protocol, a new "ContentGroup" element is added in requests, i.e., CONNECT and FIND, if the request includes a content scope. In addition, this new element is also added to the "StatisticsGroup" element for containing chunkmap information in STAT_REPORT messages. To enable the BF-based schemes to be used, the relevant interactions are described as follows:

   * (b1)In response to a CONNECT(join)/FIND request from a peer, the tracker may accompany the returned peer list with each recommended peer's BF-formed chunk availability bitmap, as the initial guidance for the requestor to start looking for neighbors in the same swarm. The additional cost for bearing the chunk-level availability information is constant (O(m)) for each peer in the returned peer list.
   * (b2)STAT_REPORT: Peers use the BF(S,m,H) algorithm for compressing the subset of locally stored and integrity verified chunks (set S) in terms of a given swarm-ID, whenever reporting or updating its chunk availability information with the tracker. As the length of each BF-bitmap is constant (O(m)), this will greatly reduce the tracker's resource expenditure in communicating and storing such information for a large peer population.

c. Integration to the peer protocol [I-D.ietf-ietf-ppsp-peer-protocol]:

   * (c1)HANDSHAKE: Peers exchange their local settings for the chunk compression schemes via HANDSHAKE messages.
   * (c2)HAVE: Peers use the BF(S,m,H) algorithm for compressing the subset of locally stored and integrity verified chunks (e.g. set S2 for Peer 2 in Figure 3) in terms of a given swarm-ID, whenever sharing its chunk availability information with another peer. The length of each BF-bitmap is constant (O(m)).
   * (c3)REQUEST: For a downloading peer to decide which neighbor to request for a given chunk_ID s, it uses the member query algorithm MT(s,bf,m,H) on each neighbor's BF-bitmap bf. The computation cost for this member check is constant (O(m)). It is also optional for a requesting peer to use BF-bitmap to indicate its data request to another peer, if needed.

3.3. Summary

3.3.1. Base TP protocol

In view of the wish that the "configuration processes for the PPSP Tracking facility, the service Portal and content sources are not standardized, enabling all the flexibility for implementers. But as there could have been different options to realize the chunk availability information (i.e. the chunk bitmap), it should be systematically configured so as to be mutually understandable to both parties. Therefore, we propose to add the following sentence to [I-D.ietf-ppsp-base-tracker-protocol] Section 5.1.1 "Installation and Initial Setup": "In case of a peer or the tracker wishes to exchange further information about the available peers in a flexible way, e.g. the chunk availability information of a specific peer in the same swarm could be represented in a various ways, there should be a way of negotiation/indication about the specific method/parameters in use, e.g. in the MPD file downloaded by the requesting peer from the web portal."

3.3.2. Extended TP protocol

The "ContentGroup" element in STAT_REPORT message has been extended to contain contentmap information, as shown in Figure 4).

Element Name or Attribute Name	Use	Description
ContentGroup	0...1	Provides information on content.
CAM	1	Describes the chunk addressing method of this content. The value is identical with the value of Table 6 of [I-D.ietf-ppsp-peer-protocol]
Representation	1...N	Describes a component of content.
@id	M	Unique identifier for this Representation.
SegmentInfo	1	Provides segment information.
@startIndex	M	The index of the first media segment in the request scope for this Representation.
@endIndex	OP	The index of the last media segment in the request scope for this Representation.

Legend:
Use for attributes: M=Mandatory, OP=Optional, CM=Conditionally Mandatory
Use for elements: minOccurs...maxOccurs (N=unbounded)
Elements are represented by their name (case-sensitive)
Attribute names (case-sensitive) are preceded with an @

Figure 4: Semantics of ContentGroup.

In order to incoporate BF-based representation, we propose to

- use the "SegmantInfo" field for the BF-formatted bitmap; and
- the three attributes of "startIndex", "endIndex" and "chunkmapSize" are meaningless in this case and are allowed to be absent.

3.4. Peer protocol

In peer protocols, the bitmap representation schemes are called "chunk addressing schemes", whose choice and configuration is included in the protocol options information exchanged between peers in communication initiation via HANDSHAKE messages. As shown by Figure 5), there are currently five options defined for bitmap representations.

Method	Description
0	32-bit bins
1	64-bit byte ranges
2	32-bit chunk ranges
3	64-bit bins
4	64-bit chunk ranges
5-255	Unassigned

Figure 5: PPSP Peer Chunk Addressing Methods.

After downloading and verifying a chunk, a peer updates its local chunk availability information to each neighboring peer in the same swarm using HAVE message, which consists of a single chunk specification that states that the sending peer has those chunks and successfully checked their integrity.In order to incorporate BF-based representation, we propose to

- add a defined value (e.g. 5) from "unassigned" value range of the PPSP peer chunk addressing methods for the BF-formatted bitmap; and
- use HAVE/REQUEST message to convey the BF-format array for the overall local chunk bitmap in the way as shown in Figure 6.

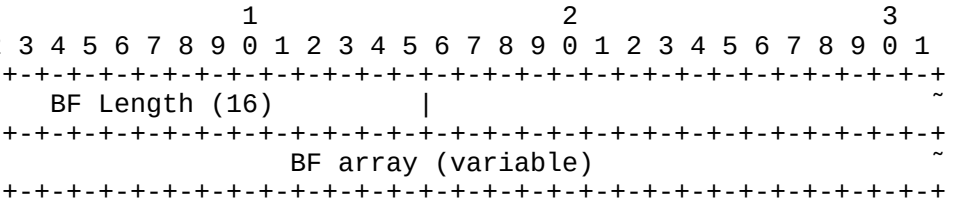

Figure 6: BF formatted bitmap.

The BF Length field contains the length of the BF-formatted bitmap that follows in bytes. The Length field is 16 bits wide to allow for flexible configuration to allow for less erroneous BF compression for large chunk groups by using longer binary arrays.

4. Open Issues

As the BF-based scheme is highly controllable and stable in terms of resource efficiency, it is viable to incorporate the contentmap information about each returned peer in the Peerlist without introducing considerable extra overhead. The current definition of PeerGroup element in the extended TP protocol needs to be extended to include "SegmentInfo" field as defined in ContentGroup element and used accordingly. However, the necessacity of introducing other peer specific information into the TP protocol via Peerlist is still subject to discussion. In particular, these info exchange is far less frequent and accurate as compared to the peer protocols's HANDSHAKE and HAVE messages.

For the sake of simplicity, the above integration with tracker protocol does not consider the case of employing more than one bitmap compression algorithms for a single swarm. Meanwhile, it is noted that there are considerable content in the peer protocol dealing with chunk addressing schemes inter-working between peers employing different schemes. However, in BF-based scheme, as for the irreversible nature of hash functions, it is not feasible to translate a BF-formatted array back into an original bitmap. Therefore, in order to keep compatible and open to peers using other addressing schemes, it is required that there be an original bitmap maintained by the local peer to allow the ability of communicating with peers using another addressing scheme.

Even if the chunk addressing scheme can be uniquely specified in a MPD file for a give swarm, there is still possibility that a peer who cannot support the relevant addressing scheme comes along and contact the tracker for joining intention. Therefore, there is need for such configuration exchange channel via TP protocol to allow mismatch detection and error handling. A possible solution could be to add an algorithm identifier attribute in the "SwarmId" field, and keep the value definition for different schemes consistent with the peer protocol's "Chunking Addressing Method" field.

5. Security Considerations

TBA

6. IANA Considerations

None.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

7.2. Informative References

[BF-analysis]
Broder, A. and M. Mitzenmacher, ""Network applications of Bloom Filters: a survey"", Internet Mathematics Vol. 1, No. 4, pp. 485-509, 2004.

[BF-bloom]
Bloom, B., "Space/time trade-offs in hash coding with allowable errors.", Communications of ACM Vol. 13, No. 7, pp. 422-426, 1970.

[I-D.ietf-huang-extended-tracker-protocol]
Huang, R., Zong, N., Cruz, R., Nunes, , and J. Taveira, "PPSP Tracker Protocol-Extended Protocol", draft-ietf-huang-extended-tracker-protocol-02 (work in progress), February 2013.

[I-D.ietf-ietf-ppsp-peer-protocol]
Bakker, A., Petrocco, R., and V. Grishchenko, "Peer-to-Peer Streaming Peer Protocol (PPSPP)", draft-ietf-ppsp-peer-protocol-06 (work in progress), February 2013.

[I-D.ietf-p2psip-base]
Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", draft-ietf-p2psip-base-26 (work in progress), February 2013.

[I-D.ietf-ppsp-base-tracker-protocol]
Cruz, R., Nunes, M., Gu, Y., Xia, J., and J. Taveira, "PPSP Tracker Protocol-Base Protocol (PPSP-TP/1.0)", draft-ietf-ppsp-base-tracker-protocol-00 (work in progress), February 2013.

[I-D.ietf-ppsp-problem-statement]
Zhang, Y. and N. Zong, "Problem Statement and Requirements of Peer-to-Peer Streaming Protocol (PPSP)", draft-ietf-ppsp-problem-statement-12 (work in progress), January 2013.

[P2P-limit]
Feng, C., Li, B., and B. Li, "Understanding the performance gap between pull-based mesh streaming protocols and fundamental limits", in Proc. of IEEE INFOCOM , 2009.

[RFC1321] Rivest, and Newport, "RFC 1321: The MD5 message-digest algorithm", April 1992.

Authors' Addresses

Lingli Deng
China Mobile

Email: denglingli@chinamobile.com

Jin Peng
China Mobile

Email: pengjin@chinamobile.com

Yunfei Zhang
CoolPad

Email: hishigh@gmail.com

Yihong Huang
Huawei

Email: rachel.huang@huawei.com