论文题目

Hierarchical Edge Caching in Device-to-Device Aided Mobile Networks:Modeling,Optimization, and Design

IEEE JSAC 2018，(SCI, CAS/JCR-1, IF: 8.085)

问题记录

1. **为什么将核心网/骨干网到边缘子网络的连接称为 backhaul（回程）？**

   网上没找到这个答案。根据个人理解，因为数据包从核心网来，到核心网去，这便是一个回程。

2. **Device-to-Device Aided Mobile Network 中 Aided 指的是什么？**

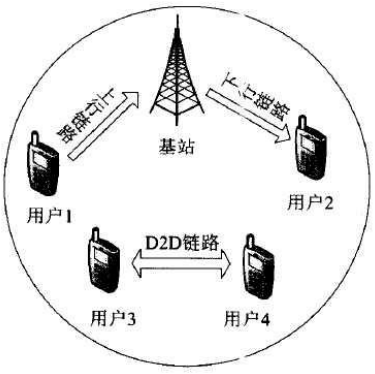
   指的是 D2D 通信技术在移动网络中的辅助作用：提高网络频谱和降低能耗；

   补充：在传统蜂窝网络中，通信被分为了上行和下行两个链路，用户之间的直接通信是不被允许的；

   D2D communication 使一定距离范围内的用户通信设备直接通信，以降低对服务基站的负荷，提高网络频谱和降低能耗；

   D2D 通信分为集中式控制和分布式控制。集中式控制由基站控制 D2D 连接，基站通过终端上报的测量信息，获得所有链路信息，但该类型会增加信令负荷；分布式控制则由 D2D 设备自主完成 D2D 链路的建立和维持，相比集中式控制，分布式控制更容易获取 D2D 设备之间的链路信息，但会增加 D2D 设备的复杂度；

   在本文实验中，D2D 通信采用的是集中式控制方式。



3. **为什么将 MDS 应用在 BSs 中？**

   （1） 可以增加 BSs 中的 content 多样性，以致：

   ① 增加 user device 在向 BS 请求 content 时的 cache hit ratio；

   ② 由于 content 的请求符合 MZipf 分布，所以这能使得 BS 响应 user request 负载均衡。

   （2） 提高 BS 之间的协作性，以致提高 BSs 的信道利用率（BSs 采用全连接的方式进行连接，提高整体 BSs 的信道利用率可以降低编码块在 BSs 的排队时延）。

4. **若缓存不命中，与直接从核心网下载的速度比较如何？**

   （1） 若两级缓存（D2D 缓存+BS 缓存）不命中则需要再去请求核心网下载 content，则比直接去核心网下载的速度要慢（即$T_{两级cache\ miss} > T_{直接核心网下载}$，$T$ 为整个缓存系统对用户请求 content 的响应时间）；

   （2） 若 D2D 缓存不命中而 BS 缓存命中，则需要进行实验测试出平均的 content 请求的响应时间，才能下定论，本文未进行该类测试；

Note:本文主要考虑的是 Backhaul 链路和 BS-BS 链路的经济和能耗成本，且因为 Backhaul 上的每单位流量负载的平均成本远大于 BS-BS 的，所以本文目的是最小化 Backhaul 的流量负载。本文未比较用户请求 content 的响应时间。

5. **该架构使用纠删码的流程？**

本文的两级缓存系统架构在 BS 层（Upper level）上采用了纠删码策略。使用纠删码的流程可分为两个子流程：（1）编码流程（2）解码流程；

(1)编码流程

MNO core 管理和决策 content 在 Upper level 的放置，即在 MNO core，content 被编码成 M 个具有 MDS 性质的编码块，分布在 N 个 BS 中。但是本文未明确 M 与 N 的大小关系，即一个 BS 中可存多个 k 个编码块（1≤k，k≤M,但这两个等号不能同时成立）；另外，本文实验假设了每个 BS 中的 cache 空间大小是相同的，但若为了最大化 content 的多样性，则分布在 BS 中的编码块数量应尽可能相同。

(2)解码流程

当 D2D 层（Bottom level）缓存不命中时，用户向 Local BS 发出请求 content 消息，才会进行此处的“解码流程”。

S1:Local BS 查找自己的缓存，若能满足则直接响应（通常情况下 Local BS 是不会缓存了足够多恢复出 content 的编码块）；

S2:若不能满足，则 Local BS 会向 MNO core 查询 content 编码块在其他的 BS 中的位置，MNO core 调度有 content 编码块的 BS，传送 content 编码块给 Local BS，若能恢复出 content 则 Local BS 将 content 回送给用户设备；

S3:若 Local BS 仍不能恢复出 content，则 Local BS 向核心网请求足够多的编码块。当 Local BS 恢复出 content 后回送给用户设备。

注意：

会进行 S3 的原因是某些 BS 中的 content（记为 f）的编码块被替换了而一些 BS 的 f 的编码块未被替换，于是存在了全局的 BS 也不能恢复出 f 的情况。

6. **为什么说在 BSs 层使用纠删码能增加内容的多样性？**

这里所说的增加内容的多样性指的是：使用了纠删码之后,一个缓存空间有限的 BS 可以缓存更多由不同的 content 编码成的编码块。

7. **分层缓存内容的关系，互斥？相同？**

若忽略掉在 Upper level 使用纠删码所带来的校验块所带来的存储开销，则可以做以下假设。

假设 1：①该分层缓存系统初始缓存状态为空；②Upper level 总 cache 大小>Bottom level 总 cache 大小，记为$S_{upper} > S_{bottom}$。

那么，Upper level 缓存的内容包含 Bottom level 缓存的内容。因为 Bottom 缓存的内容是来自与 Upper level 的，而且 Upper level 不会自己去更新缓存内容，且 $S_{upper} > S_{bottom}$。

假设 2：①该分层缓存系统初始缓存状态为空；②Upper level 总 cache 大小<Bottom level 总 cache 大小，即$S_{upper} < S_{bottom}$。

那么，Upper level 缓存的内容包含于 Bottom level 缓存的内容。因为当 Upper level 缓存内容达到上限后，Bottom level 请求 Upper level 未缓存的内容，则存在这种情况：Upper level 的缓存内容被替换，而由于$S_{upper} < S_{bottom}$, Bottom level 的缓存未被替换。

8. **工作场景**

本文的应用场景是缓冲相当长时间内流行度不发生变化的 content，如 video、music

等。

9. **隐私保护问题：在 D2D 层面，设备端缓存是否可能侵犯他人的隐私？**
   从内容的角度来看，这不侵犯他人的隐私，因为 content 的内容都是网络上的资源；从用户偏好的角度来看，则可能侵犯分享内容的用户的隐私。
10. **在 D2D 层面，将整份 content 缓存到 user device 中，是否存在 content size > cache size 的情况？又怎么做处理？**
    确实存在着 content size>cache remainder size 的情况。记 content i 的用户偏好和内容流行度总分为Score_i，那么在这种情况下则将 Score 最低的 content 替换出去。
11. **BSs 层面采用全连接方式是想告诉我们什么呢？**
    由于在 BSs 层采用的是纠删码恢复 content 策略，用全连接的方式，使得 BSs 之间的通信次数减少，也可以减少 content 恢复过程中的带宽，可以更加快速地在 local BS 中恢复出 content。
12. **本文工作的实验是部署在这些 BSs 等物理设备上的还是纯仿真实验？考虑我们所拥有的资源，自己能不能做？**
    本文工作的实验是一个纯仿真实验，用 python 自建仿真器。除了 Xender trace 数据集的收集/捕获，我们是可以复现这个实验的。
13. **工作流程**

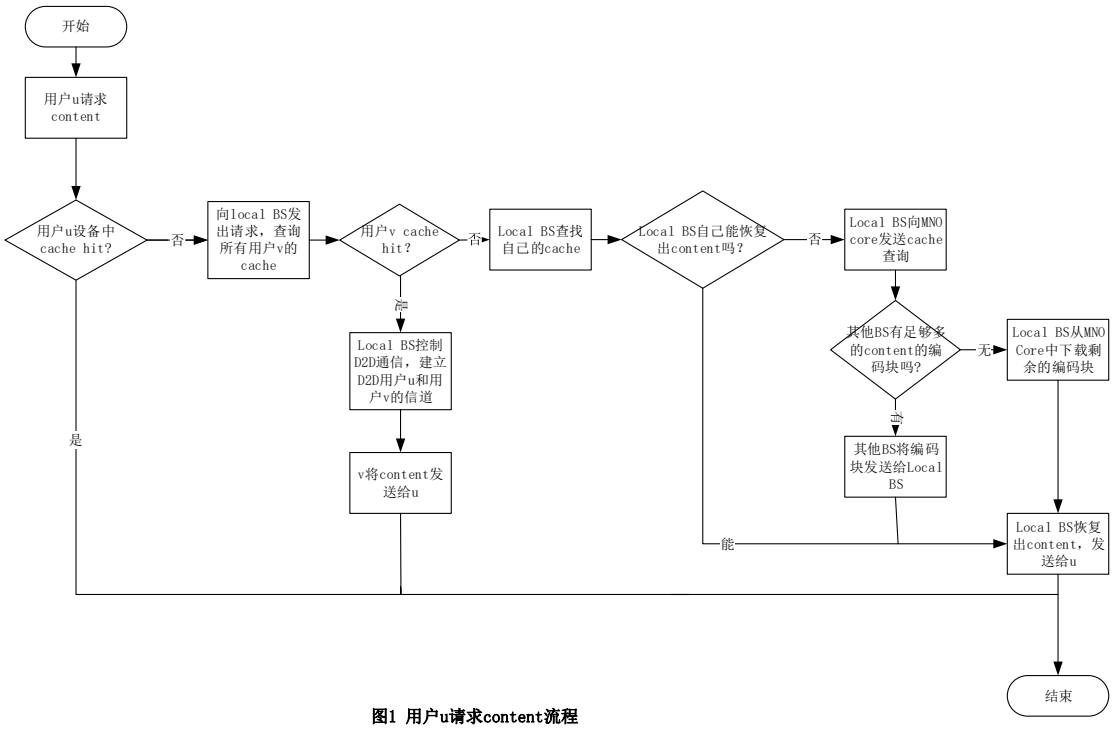
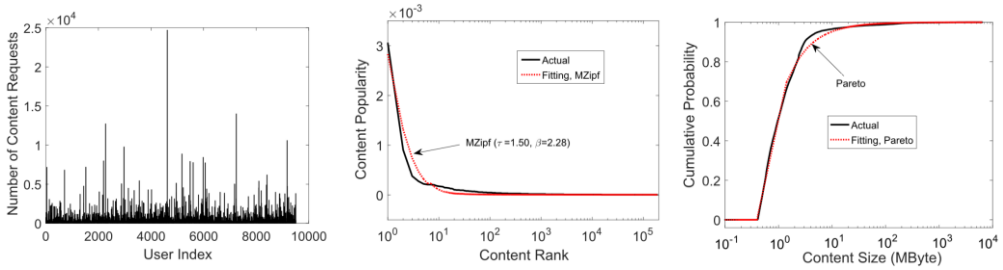


图1 用户u请求content流程

14. **实验数据的展示**

（1） 实验的数据集是：2016 年 2 月份的 Xender 的跟踪，包含了 9514 台用户设备，188447 份 content 文件，2107100 次用户请求。

（2） 对数据集进行了分析，结果如下：



(a) Content requests of users (b) Content popularity (c) Content size

以上三个图分别表明了：用户的请求次数分布均匀、content 流行度符合 MZipf 分布、content 的大小呈 Pareto 分布（即二八定律，此处指小的 content 更多）。这也验证了实验的对于内容流行度呈 MZipf 的假设是合理的。

(3) 性能提升

本文工作目标:

降低在 backhaul 网络上的流量负载，以降低所带来的经济成本.

实验设置:

所有的用户设备缓存大小相同，固定为 1GB ； BS 数量为 10 个，编码块大小为 64KB，BS-BS 链路带宽为 1Gbps，所有的 BS 缓存大小相同。以下实验研究随着 BS cache size 增大，四个性能表现（缓存系统在 backhaul 流量卸载百分比、缓存系统支持请求次数百分比、缓存系统成本降低的百分比、缓存系统中 BS-BS 链路的利用率百分比）的提升比例（与无缓存系统相比）。

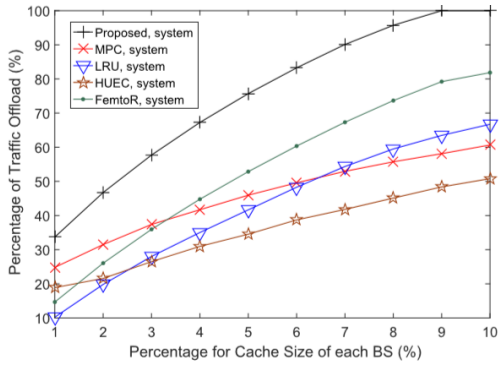
Baseline:

MPC：Most Popular Caching,该缓存系统使用本文架构，在 Upper 层和 Bottom 层均采用缓存流行度最高的 content 的 cache 替换策略；为了突出本文 cache 替换策略的有效性。

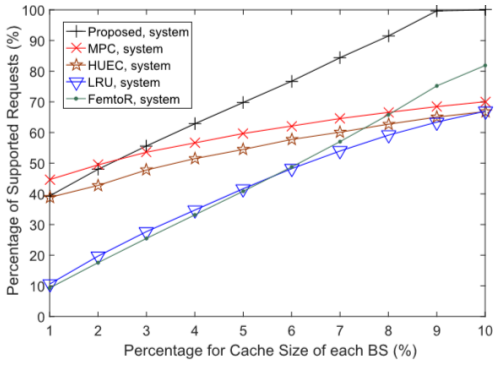
LRU：Least Recently Used,该缓存系统使用本文架构，在 Upper 层和 Bottom 层均采用 LRU cache 替换策略；为了突出本文 cache 替换策略的有效性。

HUEC：Hierarchical Uncoded Edge Caching，将其缓存系统架构修改成本文中的两级缓存架构，但在 Upper level 不对 content 进行编码，使用其论文中的 cache 替换策略；为了突出在 Upper level 使用纠删码的有效性。

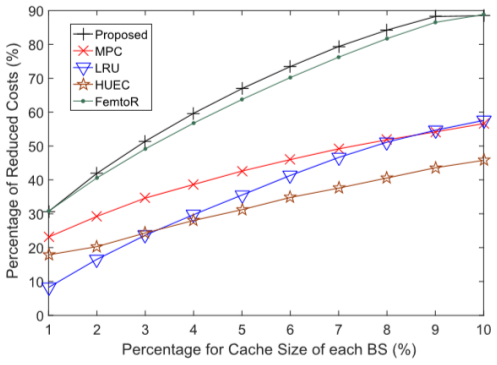
FemtoR：Revised FemtoCaching，该缓存系统使用本文架构，但不使用本文中的 D2D sharing 机制，且只在 upper level 对 content 进行编码，使用其论文中的 cache 替换策略。为了突出 D2D sharing 的有效性。



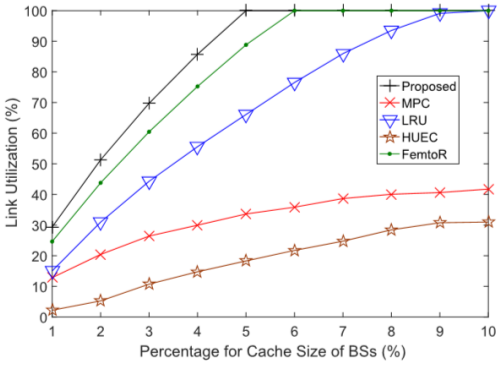
(a) Percentage of traffic offload



(b) Percentage of supported requests



(c) Percentage of reduced costs



(d) Link Utilization

以上四个图，横轴表示单个 BS 占数据集中 content 总大小的比例；纵轴分别表示：与无 cache 的系统相比，缓存系统在 backhaul 流量卸载百分比、缓存系统支持请求次数百分比、缓存系统成本（依据 backhaul 和 BS-BS 链路上的流量负载计算）降低的百分比、缓存系统中 BS-BS 链路的利用率百分比。

实验分析：

图 a 表明本文方法优于所有的 baseline，因为本文方法是最小化了（由于是 NP-hard 问题，故精确地表述是求得了最小 backhaul 流量的次优解），即尽可能地查找整个缓存系统中的 content 或者 content 编码块。

图 b 中，MPC 方法在横坐标为 1 时，是优于本文方法的。因为 Content Popularity 是根据采样时间内 User Request 统计得来的，故当 BS 不能比较好发挥出其缓存作用时（即 BS 缓存空间较小时），MPC 能达到最优的 Supported Requests 指标；但当 BS 逐渐增大时，本文方法便超过了 MPC。

图 d 中，本文方法与 FemtoR 的 Link Utilization 相近，这是因为 FemtoR 利用了由于纠删码所带来的 BS-BS 之间的协作机制。

图 c 中，本文方法与 FemtoR 的 Reduced Costs 相近，这是因为 FemtoR 的 BS-BS 链路利用率高，需要从 Backhaul 下载编码块的数据量小（Costs 主要由 Backhaul 上的流量决定的）。