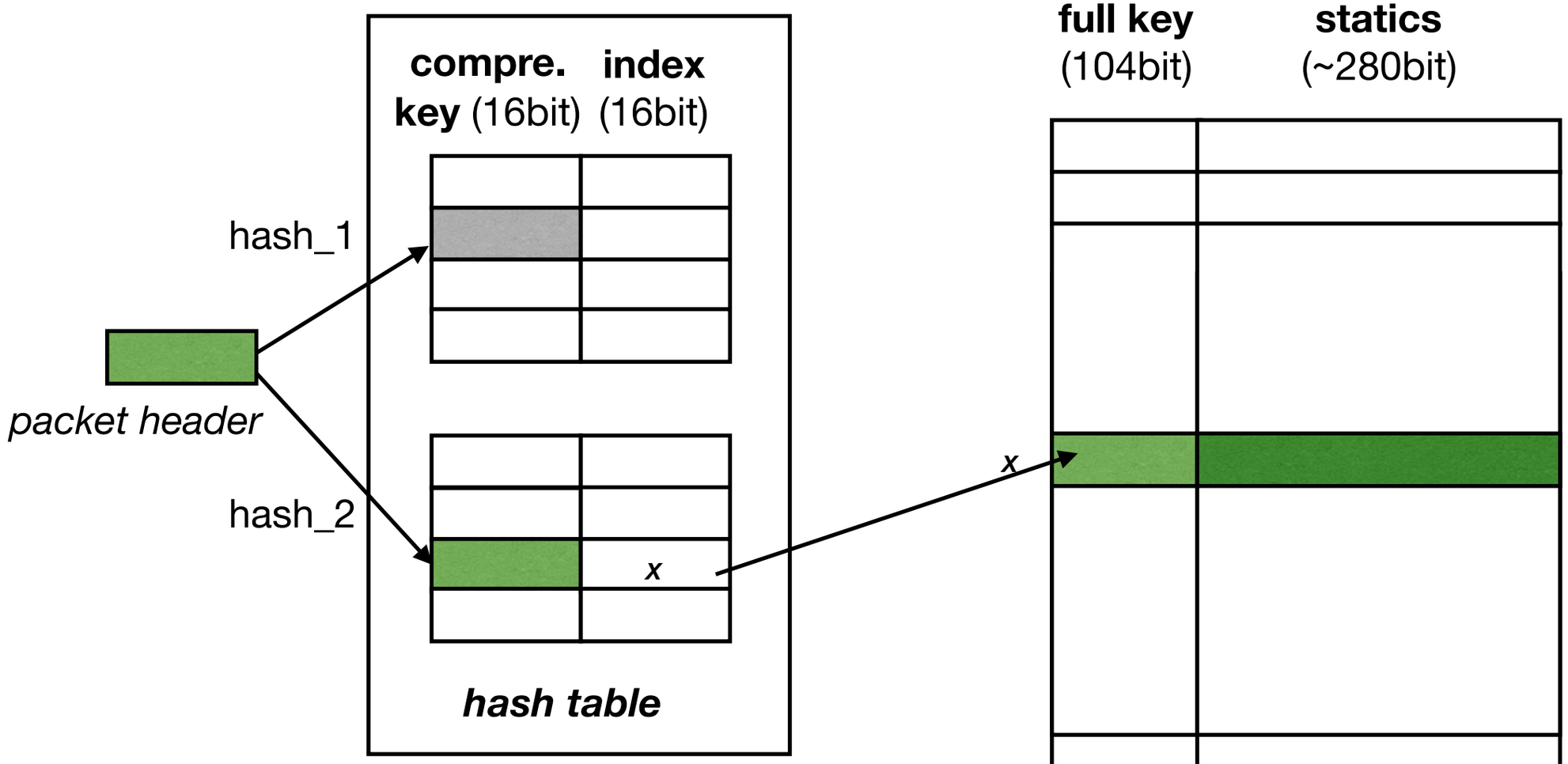
产品需求

- 实现全流信息统计
  - 流规模：**64K** flows
  - 5-tuple作为关键字信息（**104bit**），加上其它统计信息总计约**400bit**
  - 统计误差**小于1%**
- 可以使用的空间
  - 片上SRAM：**30Mb,** 访问延时为10-20ns
  - 片外DRAM（外接CPU，足够大），访问延时为200ns-500ns

1	5-tuple
2	start timestamp
3	end timestamp
4	packet number
5	byte number
6	tcp or udp flags
7	elephant flow (tag it)
8	payload lengths of the first 1 to 3 packets
9	input physical port
10	output physical port
11	none SYN (single Fin)
12	none Fin (single SYN)
13	TCP anomaly

现有算法

- 目前在SRAM上采用2-left hash实现全流统计（最大支持32K flows）

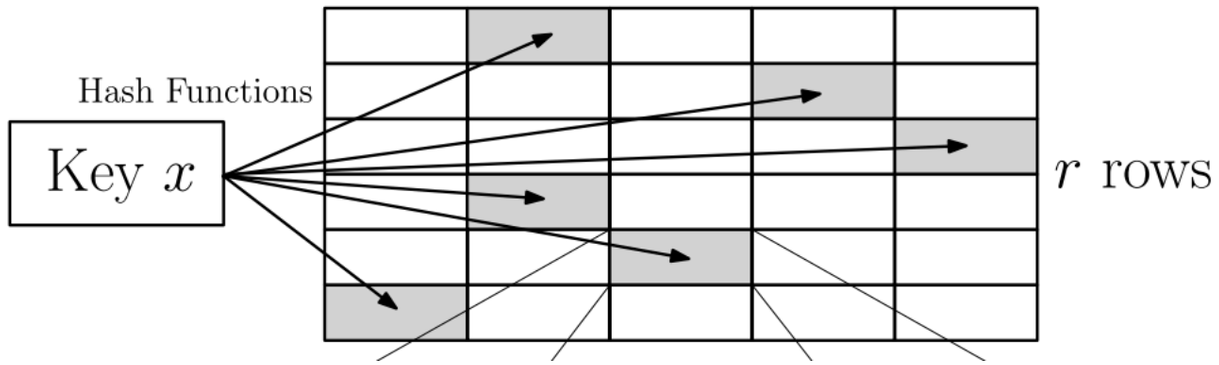


✳ hash tabel桶深为8

30Mb无法支持64K flows

算法优化(1/4)

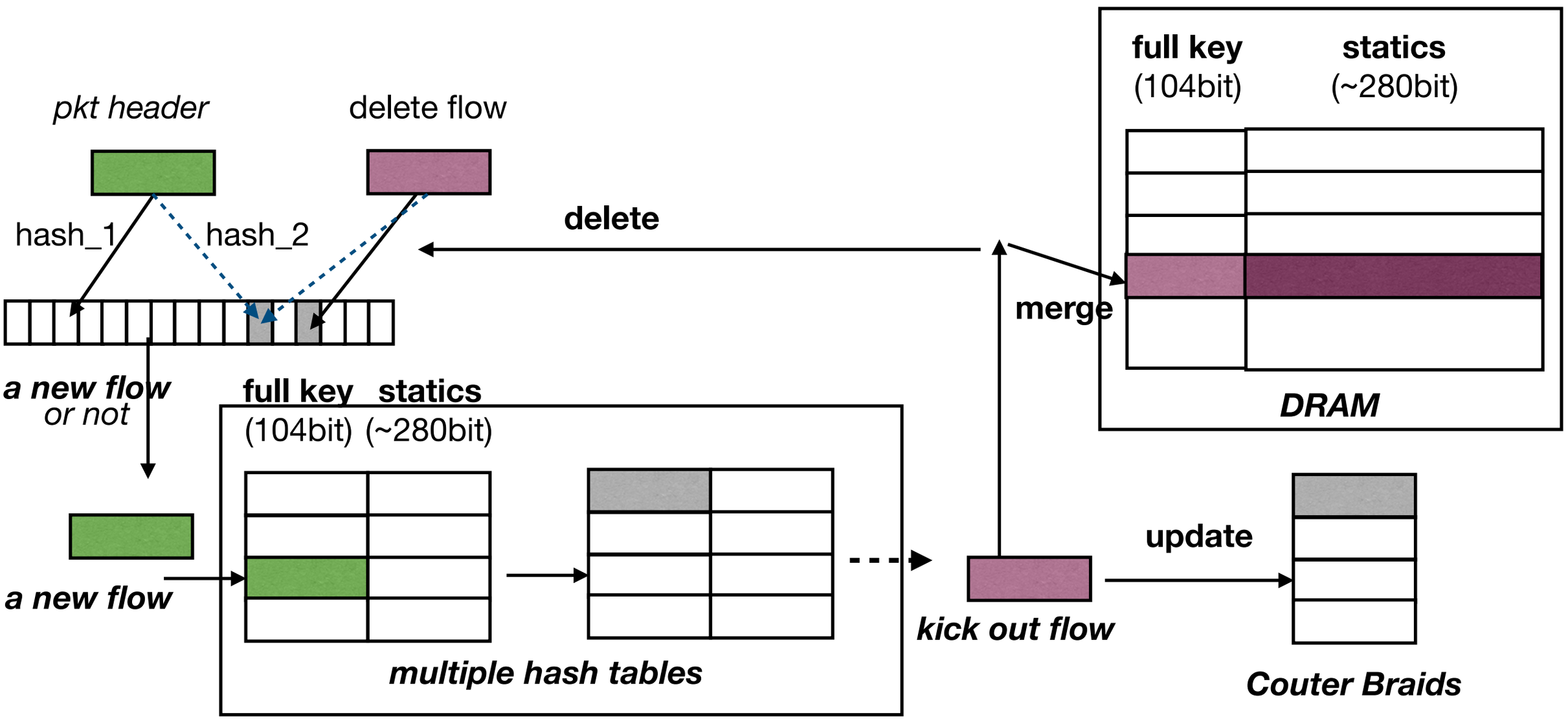
- Sketch算法（例如count-min sketch）
  - 可以节省关键字的存储，但无法恢复关键字信息，并且预估的结果具有一定误差（特别是小流的误差更大）
- sketch算法优化
  - Reversible sketch
    - 能够恢复关键字，但预测结果仍有误差
  - FlowRadar
    - 存在一定概率无法恢复关键字，且存储空间开销为x1.25，无法支持64K flows
  - Count braids
    - 能够较大概率的恢复计数信息，但无法保存其他统计信息，例如startstamp



- **实验复现发现：CB算法在3 hash，1 layer，x1.5 counter下，能够99.95%恢复count信息（流数量为64K-256K flows）**

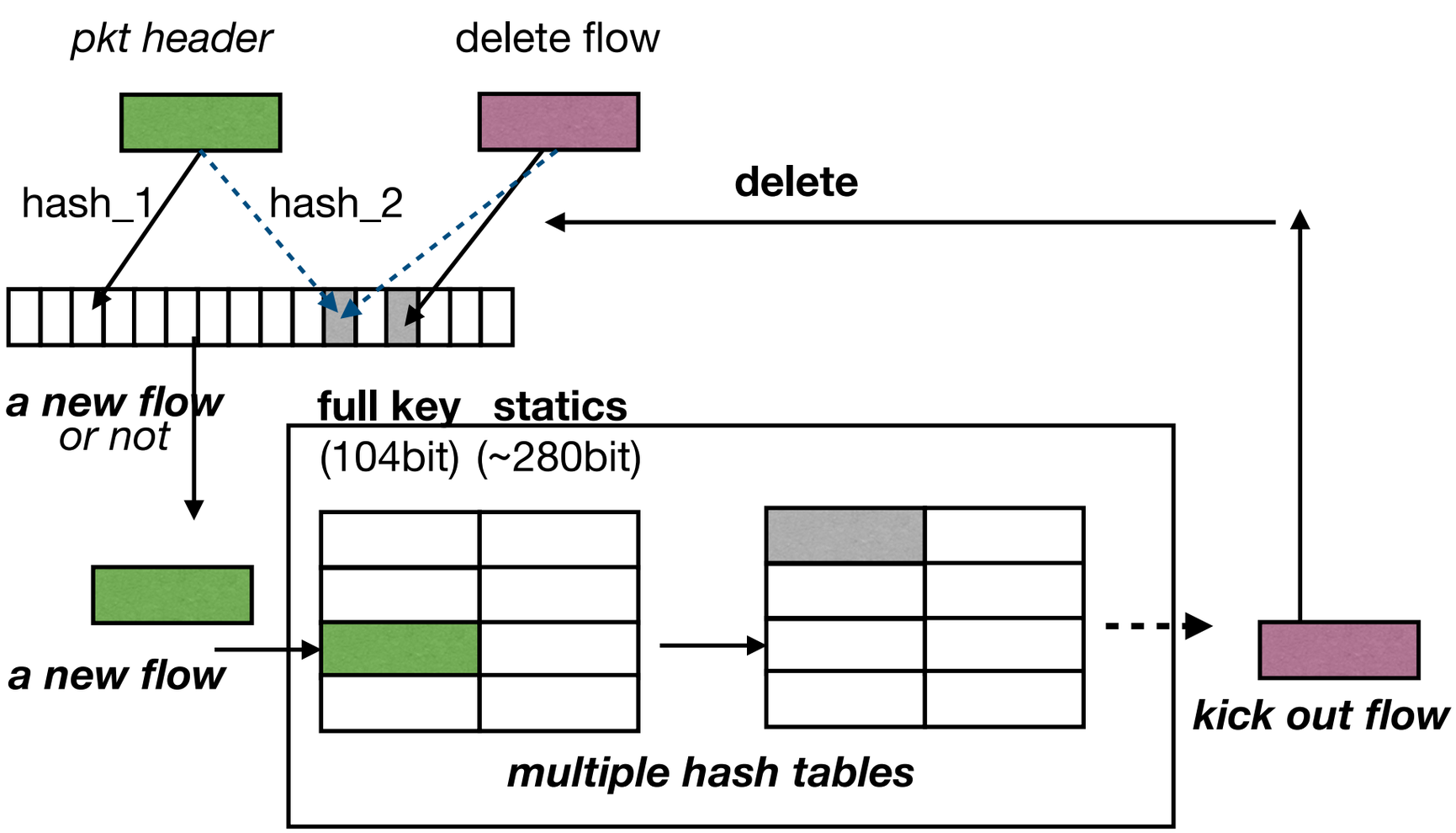
算法优化(2/4)

- 三级流水线模型
  - Bloom Filter（BF） + Space saving（SS） + Counter braids（CB）



算法优化(3/4)

- 三级流水线模型
  - Bloom Filter（BF） + Space saving（SS） + Counter braids（CB）

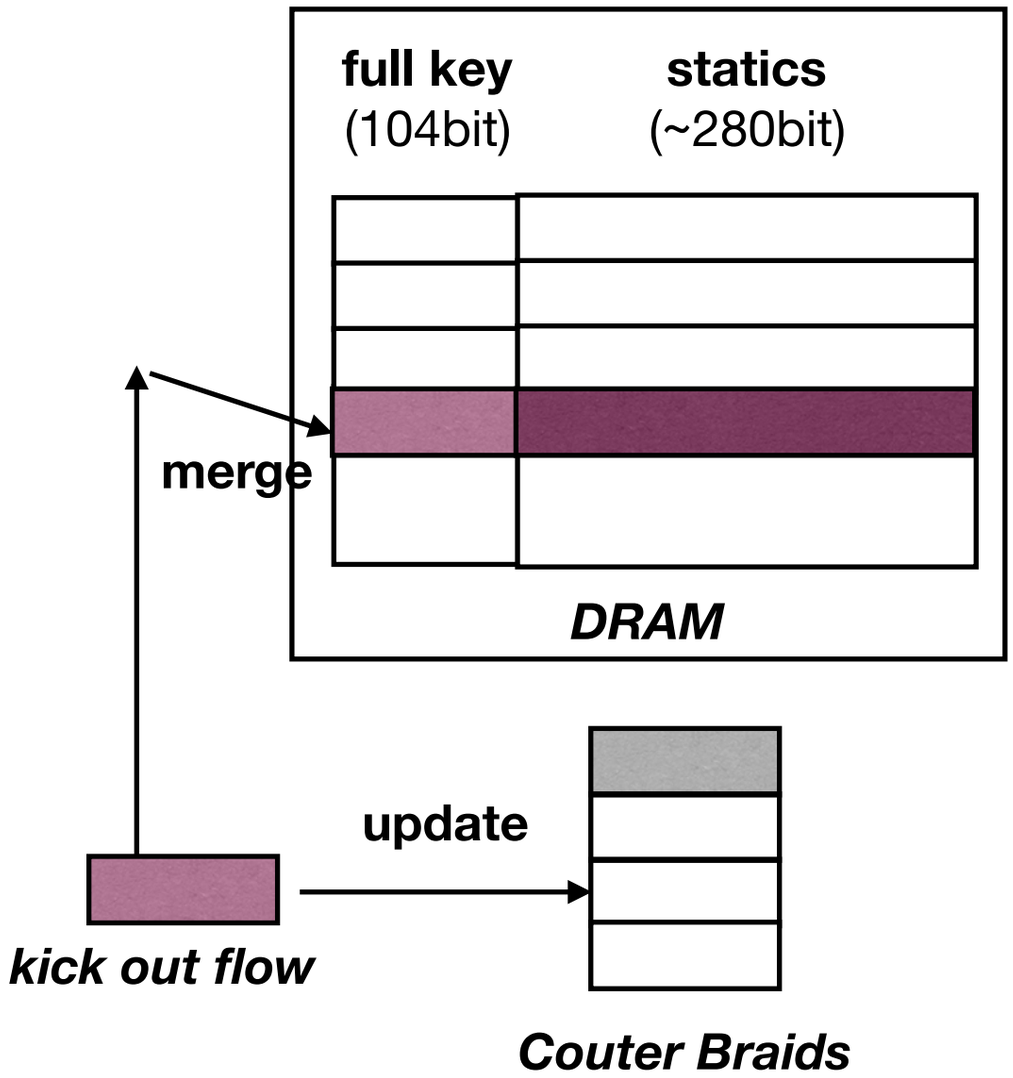


- **bloom filter的作用是保证hash tables中不同时存在相同关键字的统计信息，有利于降低表象被踢出的概率（false positive）**
- **multiple hash table则是用于实现space saving算法，利用局部最小值代替全局最小值**
- **BF采用CBF，在每次mutiHB剔除某一条流后，同时减少CBF的标志位；**

算法优化(4/4)

- 三级流水线模型
  - Bloom Filter（BF） + Space saving（SS） + Counter braids（CB）

- **multiple hash table则保留全关键字信息，以及流统计信息；当某一流被踢出后将其更新址DRAM中（merge操作）；由于流的分布（大象流占绝大部份流量，实验发现占>75%报文数），更新DRAM的频率较低**
- **将提出流的计数信息更新Counter Braids**
- **最后根据DRAM中流信息，以及Counter Braids的统计信息，恢复流的统计信息；**
- **这里DRAM仍有可能无法满足SRAM的踢出速率，因此可以再增加一个bloom Filter，用于判断该流是否已经存在DRAM中功能，优先丢弃已经存在的Flow**



网络自动化管理的需求

- 网络管理的复杂性
  - 网络拓扑复杂
  - 网络流量巨大
  - 网络规模庞大