

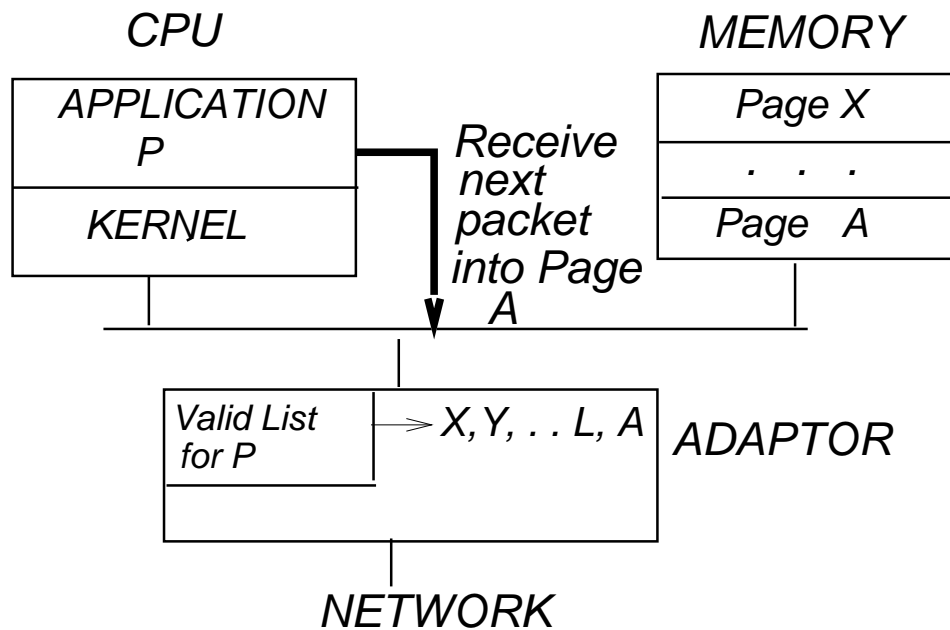
# Principles in Action

**George Varghese**

UCSD CSE  
varghese@cs.ucsd.edu

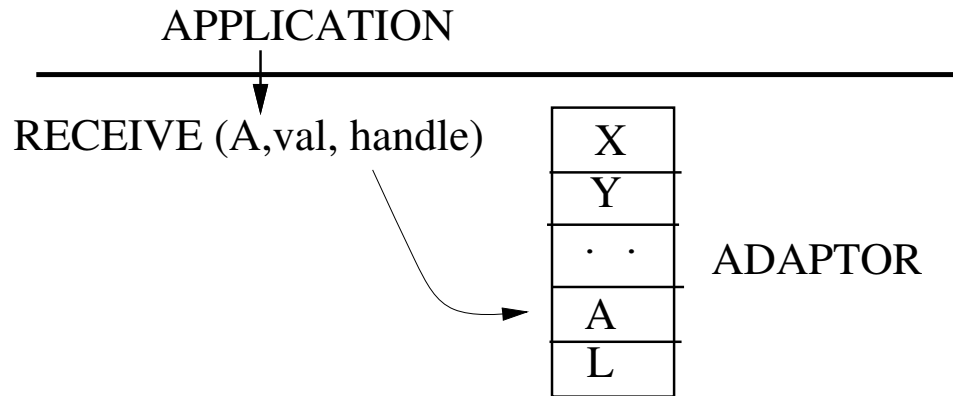
January 13, 2005

## 4.1 ADC Buffer Validation



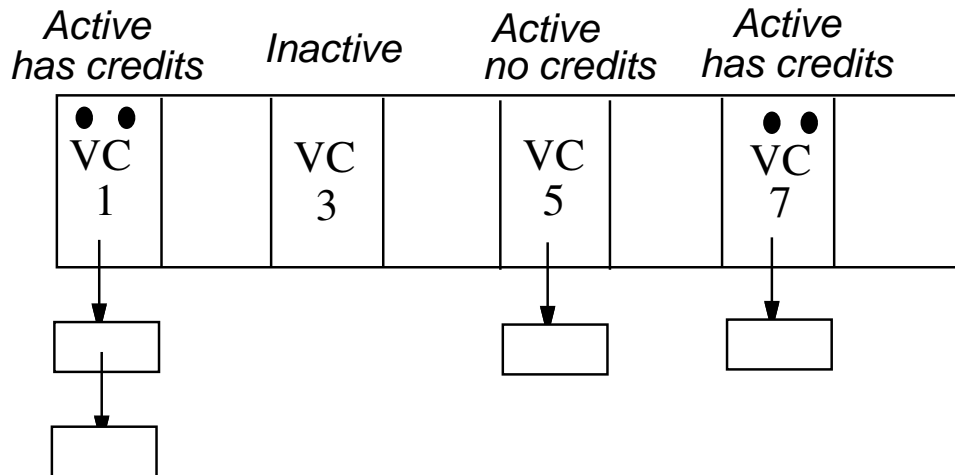
- **Problem:** When application writes to a page, adaptor must validate whether page belongs to valid page list for this ADC.
- Validation can cost  $O(n)$ , where  $n$  is size of list.

# ADC Buffer Validation



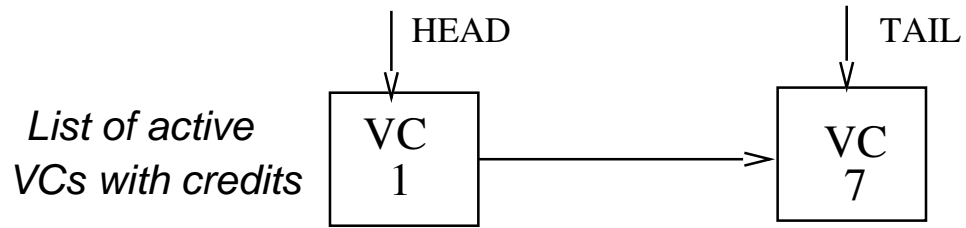
- **Solution:** Use a table of valid pages per ADC. Application passes a handle with WRITE call that can be used as an index into table. Hints versus indices

## 4.2 ATM Credit based flow control



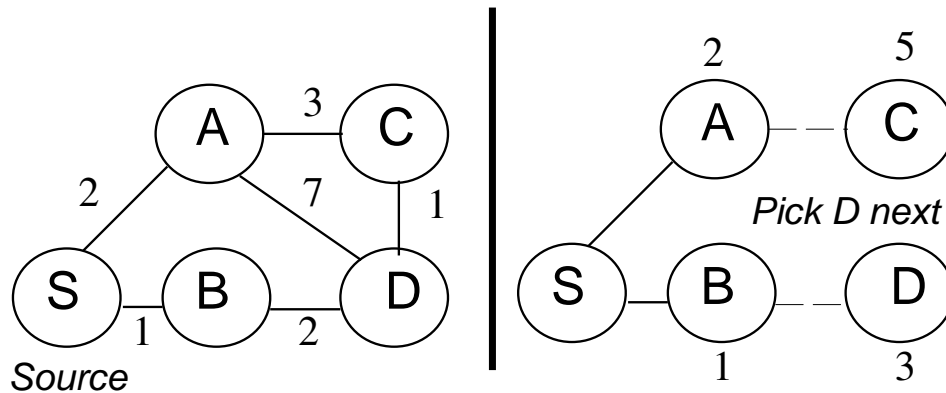
- **Problem:** Don't want to step through VCs that are inactive and/or have no credits.

# ATM Credit based flow control



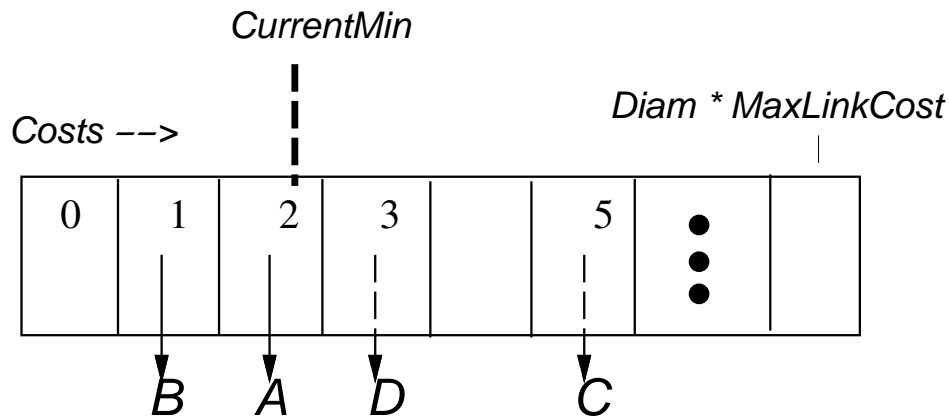
- **Solution:** Maintain a list of VCs that are both active and have credits.
- Remove VC from list after service if VC becomes inactive or has no more credits. Add VC to list when cell arrives or VC receives a credit update.

## 4.3 Route Computation using Dijkstra's Algorithm



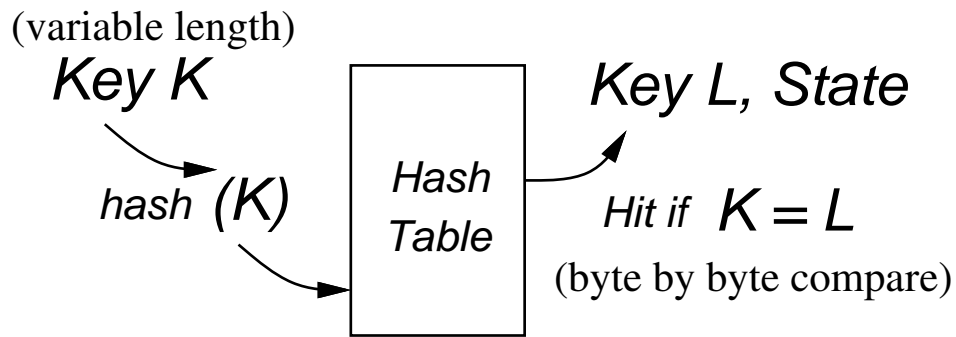
- Algorithm picks node  $D$  whose distance is closest to nodes already in tree, adds  $D$  to tree, and updates distances to  $D$ 's neighbors
- **Problem:** Requires a priority queue with  $O(\log N)$  cost for *ExtractMin*.  $\rightarrow O(N \log N)$  total cost.

# Route Computation using Dijkstra's Algorithm



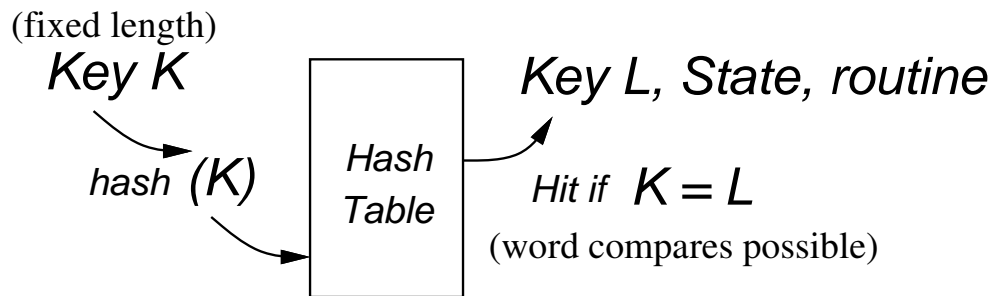
- **Solution:** Satisfies monotonicity condition so can use bucket sort priority queue. Cost includes stepping through empty buckets.  $\rightarrow O(N + Diam * MaxLinkCost)$  total cost.
- Can do with circular array of size  $MaxLinkCost$ . Why?

## 4.5 Demultiplexing in the x-kernel



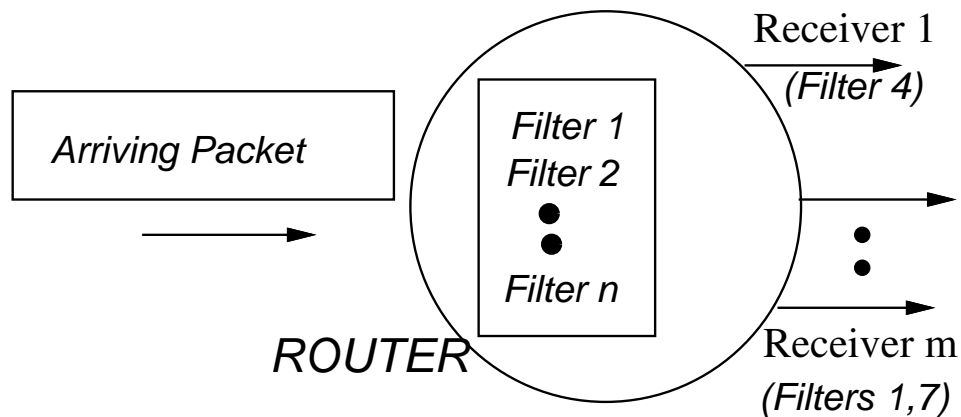
- To demultiplex, x-kernel looks up variable length protocol identifiers like  $K$  in a hash table. Must compare  $K$  to hash table entry  $L$  to check for a “hit”.
- **Problem:** Variable length keys require expensive byte-by-byte comparisons. Most common case may be word length keys.

## Demultiplexing in the x-kernel



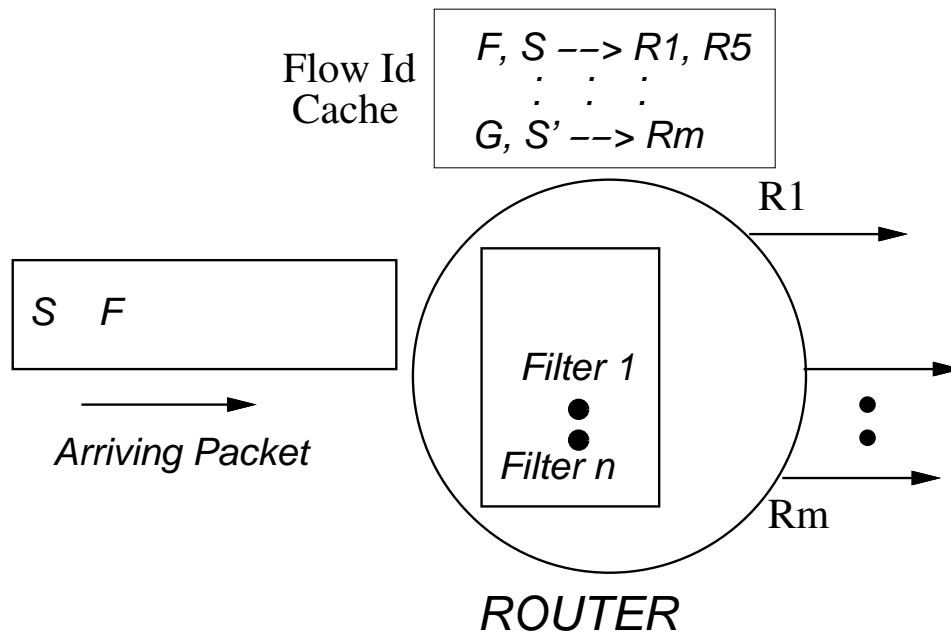
- **Solution:** Management set up time optimization. Each protocol pre-declares its identifier length so x-kernel can use specialized comparison routines for each protocol.
- A one-back cache further optimizes expected case.

## 4.7 Packet Filtering in Routers



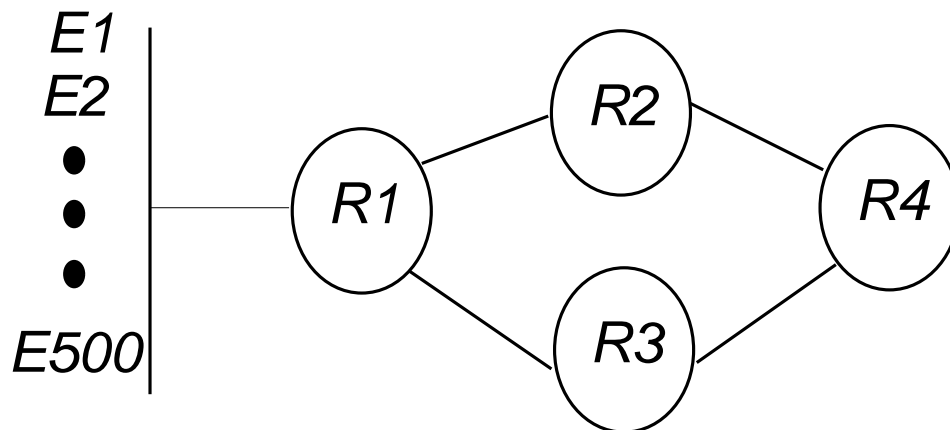
- A router implementing an RSVP-like protocol may have each receiver specify packets they wish to receive using *filters*.
- **Problem:** Each receiving packet must be matched against all filters and sent to all receivers that match. Expensive.

# Packet Filtering in Routers



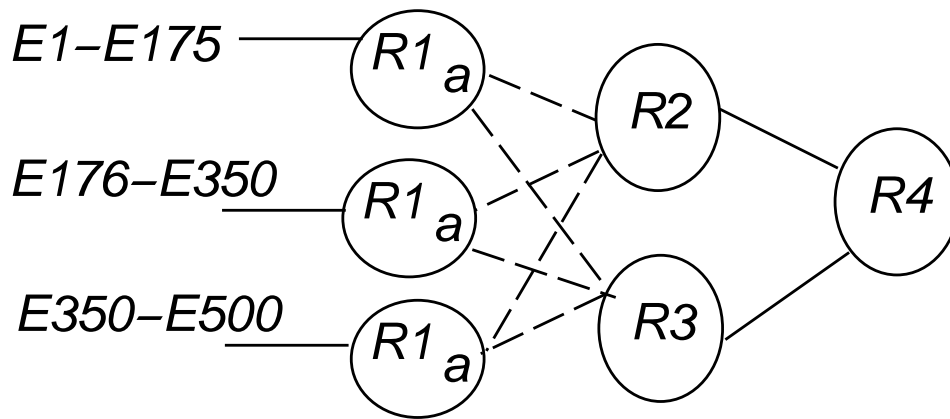
- **Solution:** Change header to add a flow ID  $F$ . Cache receivers whose filters match packet  $F$ .
- Sender must not change fields that affect a filter without changing  $F$ . Router must flush cache when filters change.

## 4.8 LSP Fragmentation



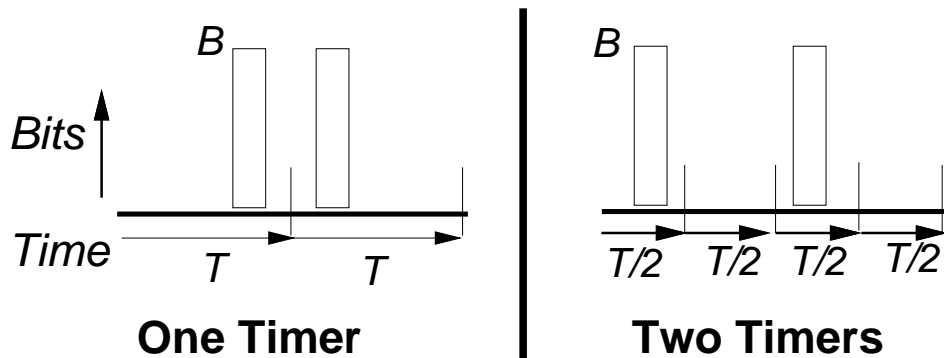
- Router  $R1$  must send a Link State Packet (LSP) listing 500 attached endnodes
- **Problem:** At eight bytes per endnode, the LSP cannot fit in a Data Link PDU. Link-by-link fragmentation and reassembly is expensive, error-prone, and slow.

## LSP Fragmentation



- **Solution:** Change protocol to allow  $R1$  to be multiple pseudo-routers  $R1_a$ ,  $R1_b$ ,  $R1_c$  and to divide endnodes among pseudo-routers.
- Pseudo-routers send unfragmented LSPs with 7 byte ID (6 byte router + 1 byte pseudo-router Id). LSP propagation treats pseudo-routers separately while route computation treats them as as one router.

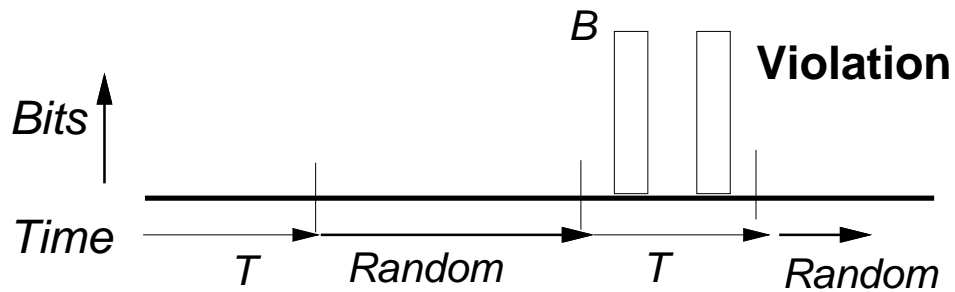
## 4.9 Policing traffic patterns



- Need to ensure that flow sends no more than  $B$  bits in any period of  $T$  seconds.

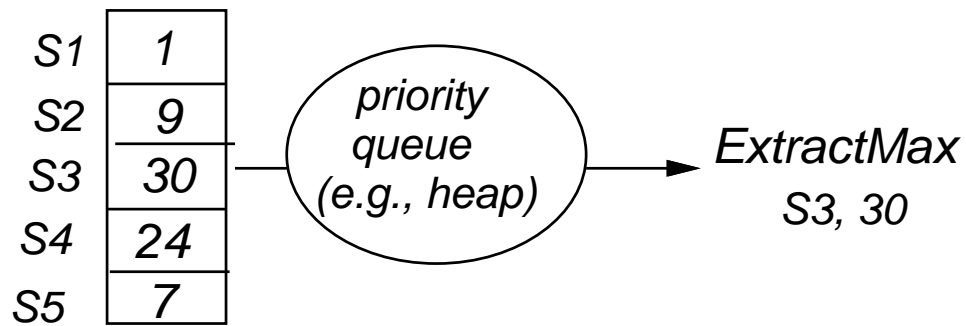
**Problem:** If we use a single timer that ticks every  $T$  seconds, and count bits, some violations will not be detected. Multiple timers/counters have similar problems.

## Policing traffic patterns



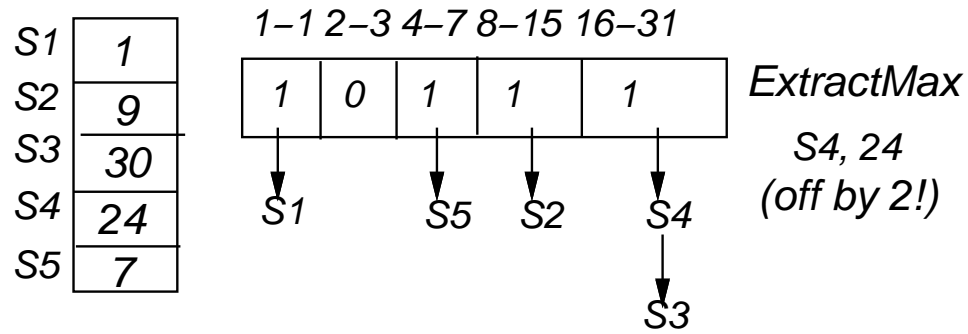
- **Solution:** Even with one timer, undetected violations can result in at most  $2B$  bits in any  $T$  seconds.
- For finer detection, use a random gap between to sample arbitrary ranges of  $T$  seconds. Can catch significant violations with high probability.

## 4.10 Identifying a Resource Hog



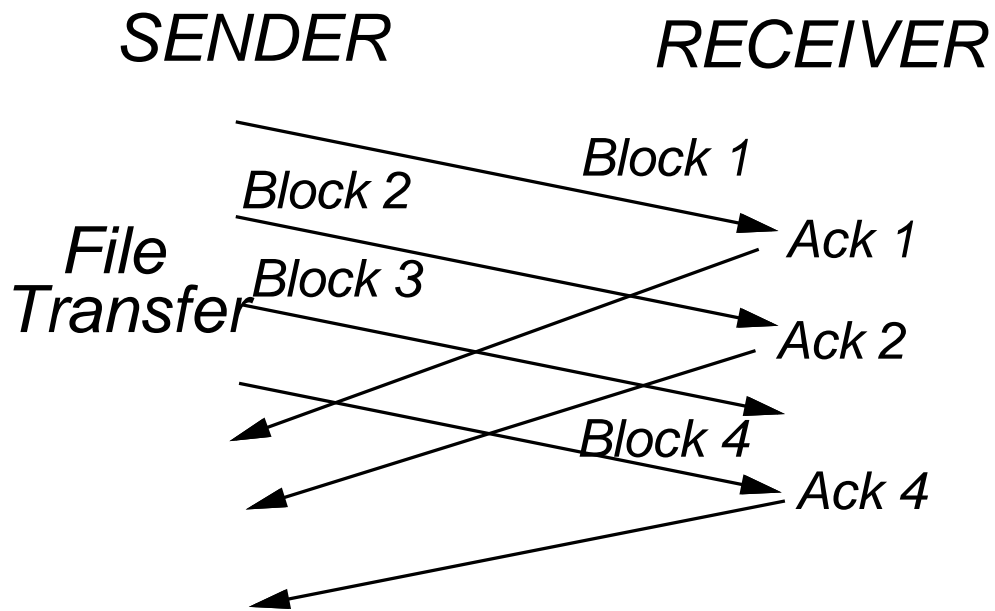
- **Problem:** Keep track of resource used by various processes (users). Want a cheap way to find the process consuming the most resource.
- Ordinary heaps may be too slow.

# Finding the Heaviest (almost) User



- **Solution:** If we settle for knowing the heaviest within a factor of two, we can use *binomial bucketing*.
- Pick arbitrary process in the highest non-empty bucket.

## 4.12 Ack Withholding

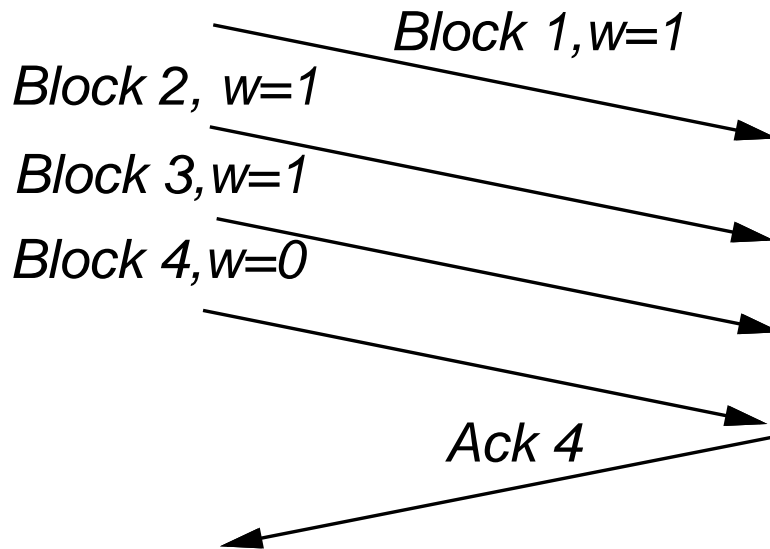


- **Problem:** Ack withholding difficult at a receiver that is not clairvoyant. Withholding in some circumstances can increase *latency*.

# Ack Withholding

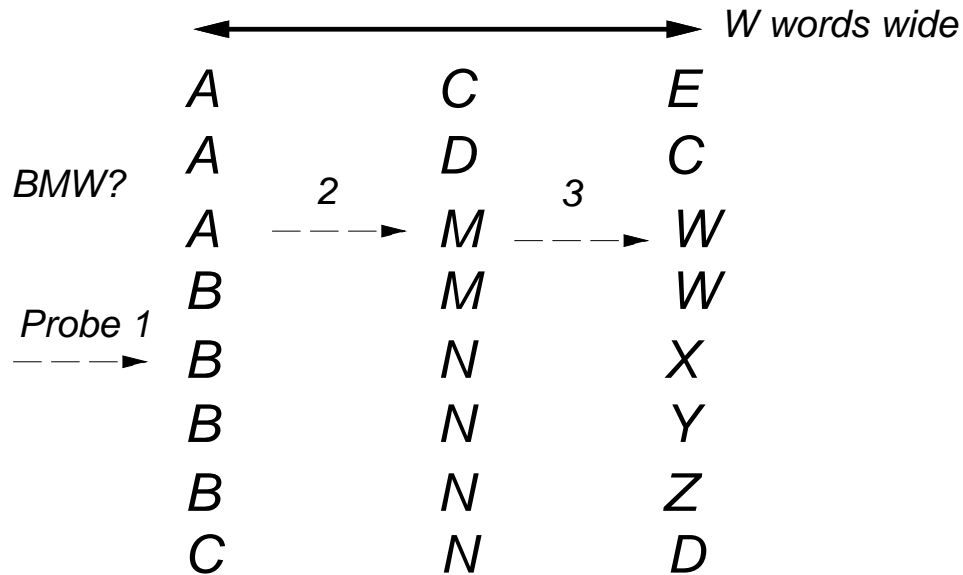
*SENDER*

*RECEIVER*



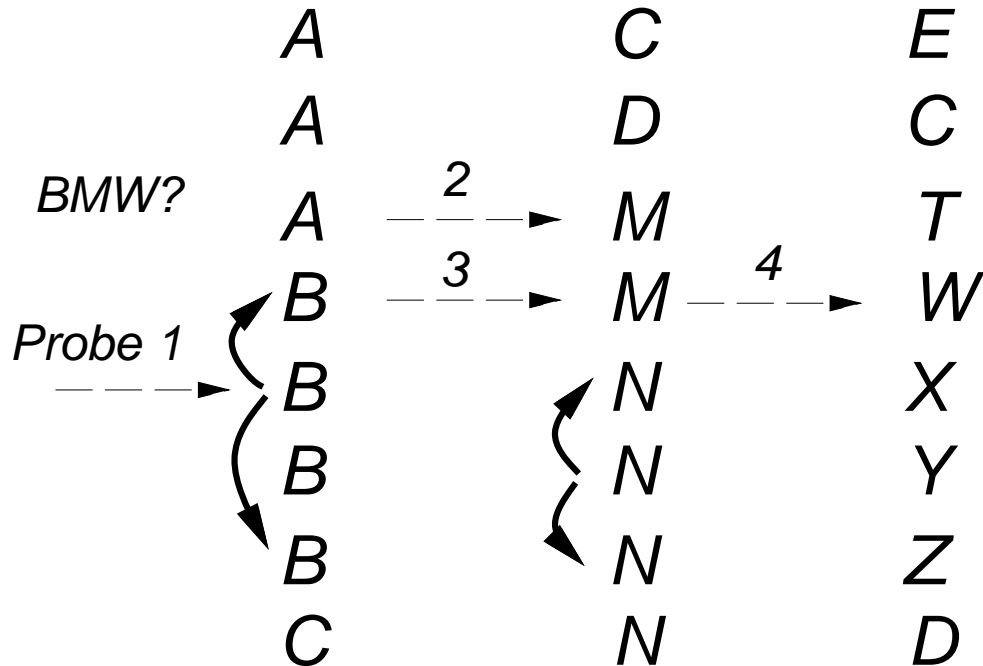
- **Solution:** Sender passes a bit  $w$  to transport when it has more data to send. When sender wants an ack, it sets the withhold bit  $w = 0$ .
- Better for sender to telegraph his intentions than for receiver to guess!

## 4.14 Binary Search of Long Identifiers



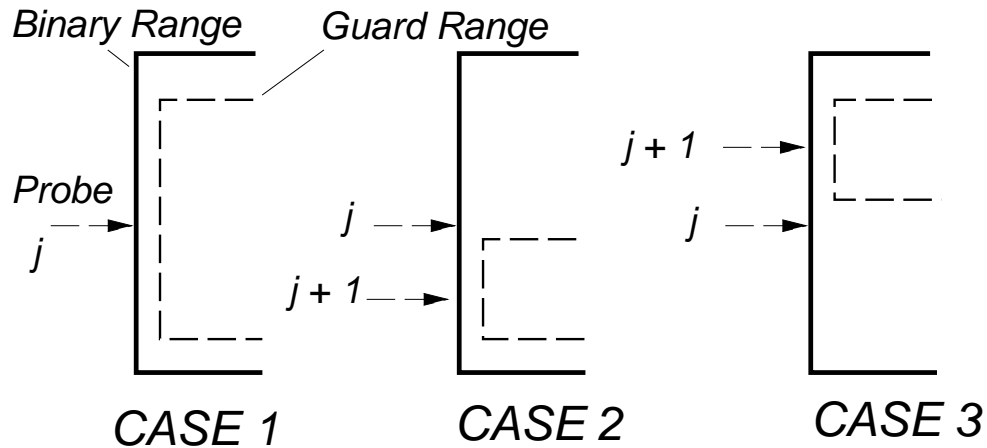
- **Problem:** Identifier is  $W$  words long. Naive binary search needs wide memory (hardware) or  $W \cdot \log N$  comparisons (software). Expensive.
- Starting in MSW and moving rightward on equality can lead to wrong identifier.

## Binary Search of Long Identifiers



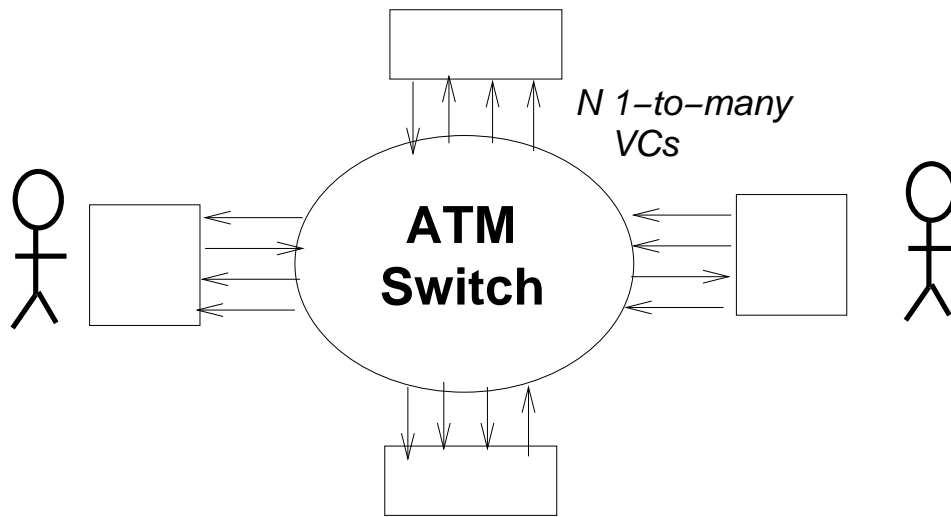
- Solution:** Add two pointers to each word  $W$  indicating range of entries that have same prefix as  $W$ . Pointers determine new range when moving rightward on equality check. Complexity =  $\log N + W$ .
- Problem:** Pointer ranges can be arbitrary. Powers of 2 easier.

# Binary Search of Long Identifiers



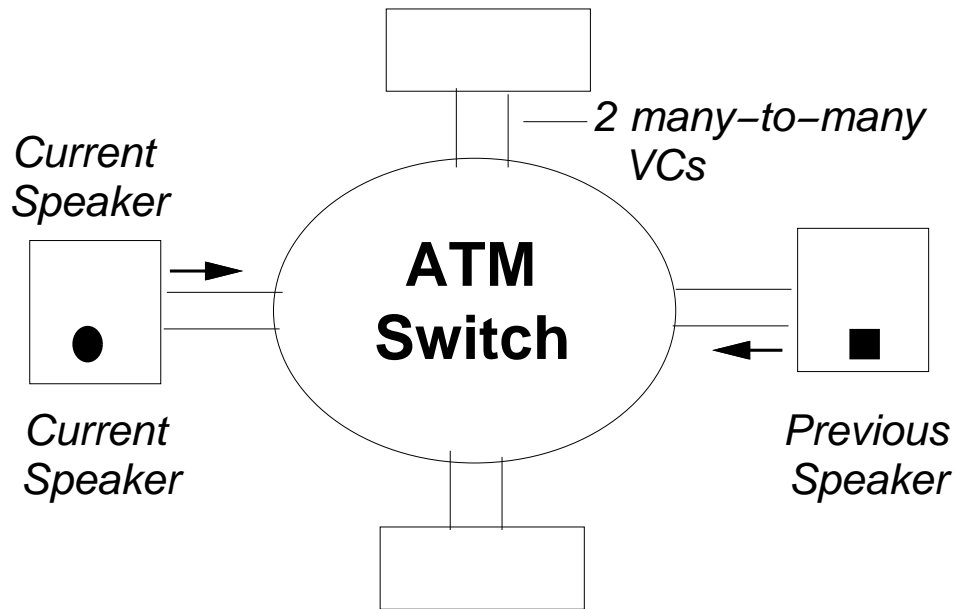
- Keep two ranges, a binary range that is a power of two, and an actual guard range that is updated when we move to the right.
- We ensure that guard range is within binary range. A form of *lazy* updating of the actual range.

## 4.15 Video Conferencing over ATM



- With  $N$  conference participants, naive solution uses  $N$  *one-to-many* VCs.
- **Problem:** Total conference bandwidth  $B$  must be split  $N$  ways. Many VCs to set up and manage for large conferences.

# Video Conferencing over ATM



- **Solution:** Use *two many-to-many VCs*,  $C$  and  $L$ . Speech detector connects video of current speaker to  $C$ , and video of last speaker to  $L$ .
- Keeping last speaker (or last few speakers) provides continuity in common case, i.e., lots of “locality” in speech patterns.