TCP Segment Caching

draft-sarolahti-irtf-catcp-00.txt

Pasi Sarolahti, Jörg Ott, Colin Perkins

ICCRG meeting

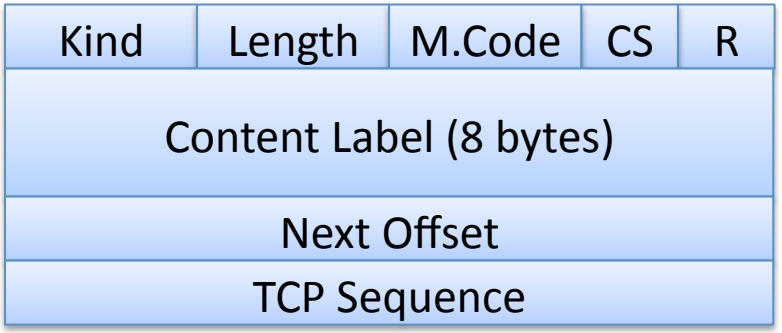
IETF-83, Paris, France

Content- and Cache Aware TCP

- Enables TCP segment caching and replication in network
  - Cachable segments are supplied with a content label
  - Common data shared between different connections
  - Cache can send segments on behalf of the sender
- Only sender TCP modifications needed
  - Works with standard TCP receiver
- Application specifies content label
  - Small API extension needed
- Example use cases
  - TCP-based media to multiple simultaneous receivers
  - Mitigating server load on sudden flash crowds

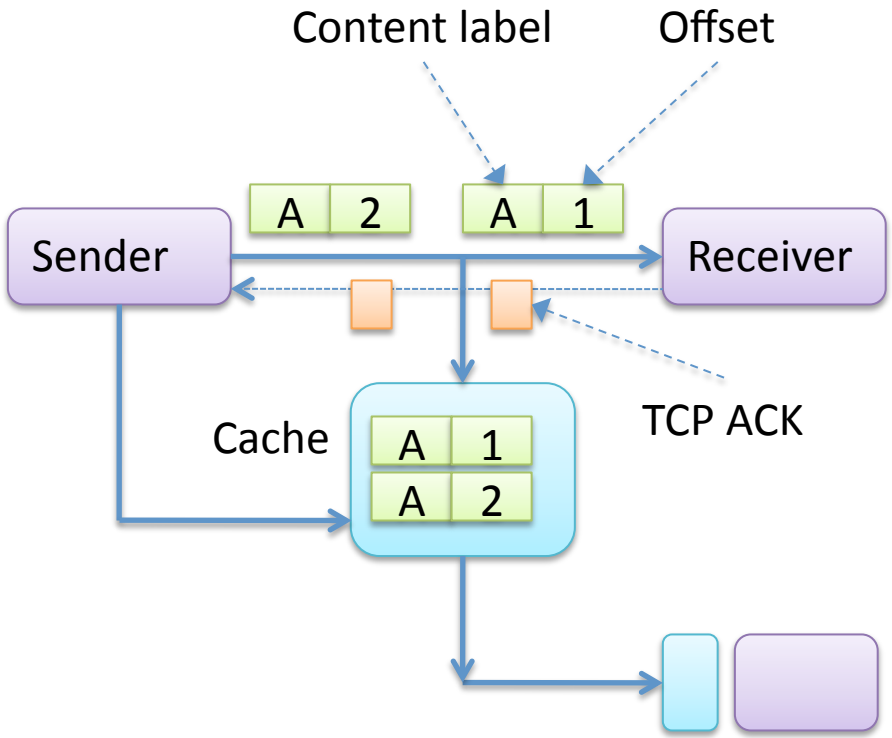
Content Label Option

- "Content Label" option in TCP data segments
  - Identifies the piece of content included with segment
  - Content object may be larger than TCP segment, therefore offset needed
- "Content Request" in TCP acknowledgments
  - To request data from cache
  - CS: number of segments that can be sent
  - TCP sequence: to be used in TCP header of cached data



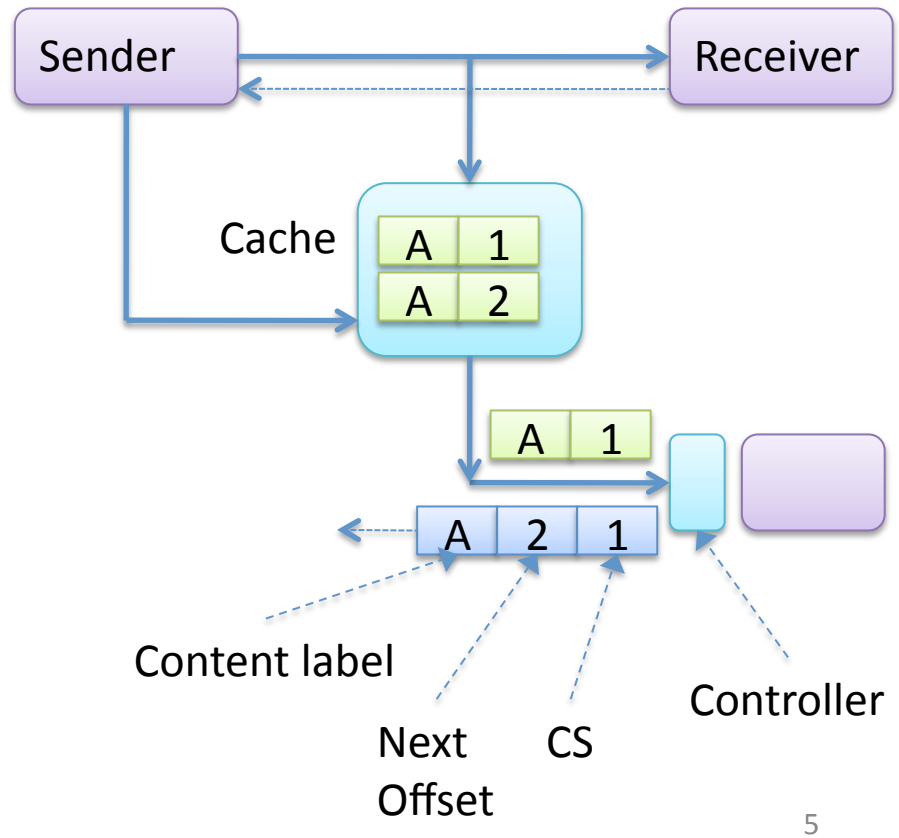
Protocol Operation

- 1) TCP sender adds Content Label option to cachable segments
  - Same connection can have non-labeled segments, and different content labels
- 2) Segment cache can store segments with content label option
  - Cache lookups happen based on label and offset
  - No per-flow state needed



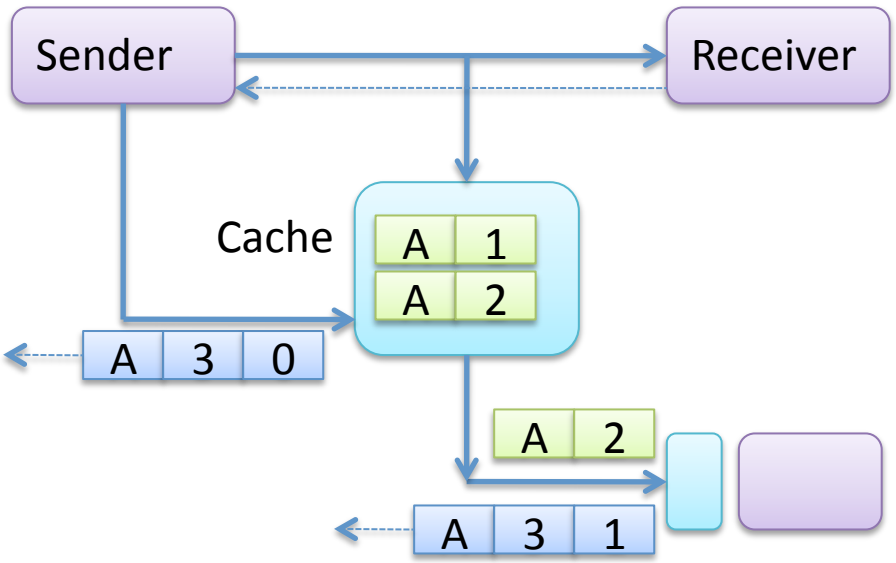
Protocol Operation

- 3) Receiver acknowledges segments normally
  - No CA-TCP support needed
- 4) Controller adds Content Request option to ACKs
  - Per-flow state for each connection
  - Can be co-located with cache or with receiver



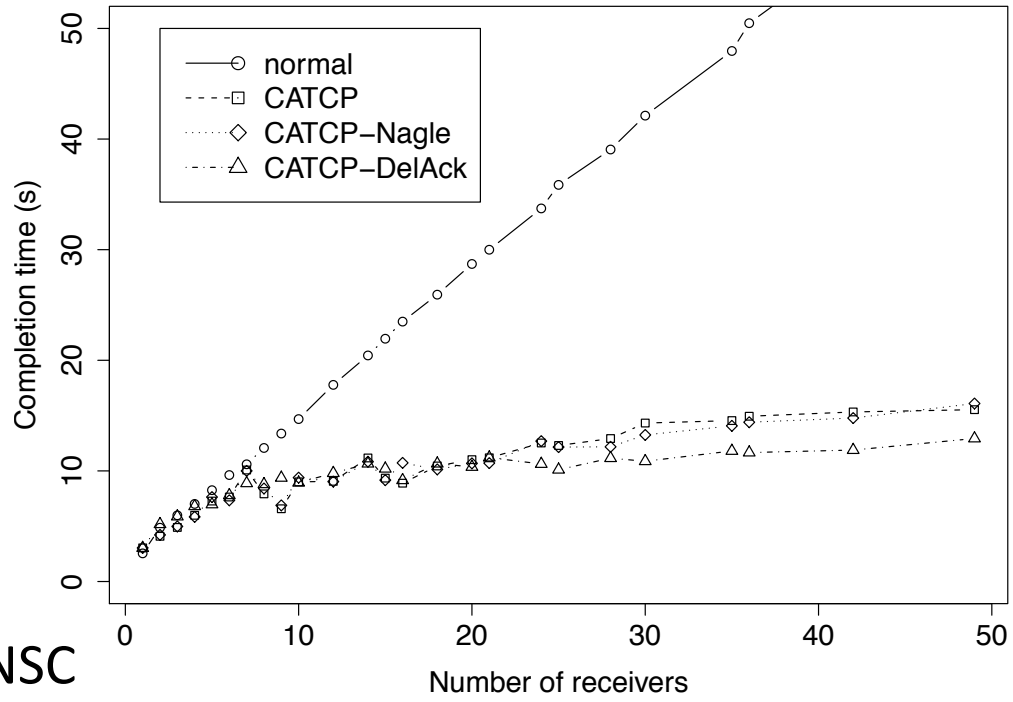
Protocol Operation

- 5) Cache can send segments based on Content Request option
  - Increases Next Offset
  - Decreases CS
- 6) Acknowledgments flow back to the sender
  - Sender does not send data if CS== 0
  - Updates SND.NXT based on “Next Offset” field



Experimentation

- TCP modifications implemented in Linux kernel
- Two cache implementations
  - Stand-alone bridge
  - Click router module
- Tests with
  - Amazon EC2 servers in different continents
  - Ns-3 simulations with NSC
  - HTTP and BitTorrent traffic



Notes and Issues

- Multiple control loops
  - Faster round-trip between cache and receiver
  - Synchronization: later flows catch up the first flow that feeds caches
- Congestion control for cached segments
  - Is simple congestion avoidance enough?
- Inconsistent segmentation may hamper cachability
  - Can be controlled (to some extent) at sender side
  - Not much can be done with re-segmenting middleboxes
- Security: attacker could send false content labels
  - Integrity checking would be needed
- Acknowledgments for unsent segments may confuse some middleboxes
- Contention of the available TCP option space

Planned Next Steps

- Improve the draft
  - In future: publish as Experimental RFC
- More experimentations
  - More diverse environments, different applications
  - Collaboration is welcome