Architectural Components and Resource Control Foundation in Data-Oriented P2P

Richard Alimi
Y. Richard Yang

IETF 77 DECADE
March 26, 2010
Basic Architecture of Data-Oriented P2P Content Distribution

• Each node connects to a set (e.g., ~30-100) of neighbors (typically do not need to be a strongly structured topology such as a tree)

• Chop data into chunks/pieces (say 256KB, or 1/3 sec. video data)

• Nodes exchange chunk availability using a data structure called BitMap or BufferMap

• Nodes request/push data with neighbors
Two Major Architectural Components of Data-Oriented P2P Content Distribution

**Topology (Peer) Management**
- Who connects to whom? Includes
  - connectivity among peers;
  - who connect to sources/super nodes/CDN.

**Chunk (Data) Scheduling**
- Who serves whom at what rates? Includes
  - A downloader requests from which uploaders
  - An uploader serves which downloaders at what rates

We can consider both components as conducting resource control on resources, including
- connection slots
- upload/download bandwidth
- storage capability
Why is BW Resource Control Important and Fundamental in P2P Systems?

• Because BW resource control is fundamental for
  – Robustness against selfish behaviors
  – Robustness against attacks
  – Construction of efficient flow distribution patterns (in particular for streaming)
Robust Against Selfish Behaviors

• P2P systems depend on user contributions

• Non-contributing users can be a serious problem
  – 70% of Gnutella users share no files and nearly 50% of all responses are returned by the top 1% of sharing hosts

• BW resource control is a major mechanism to design incentives and handle selfish behaviors
  – BitTorrent Tit-for-Tat
    • Attacked by BitTyrant
  – Provable Proportional Sharing [STOC’07; SIGCOMM’08]
Robust Against DoS Attacks

• A recent study [IMC’08] showed how to attack the Akamai streaming servers due to sharing of server bandwidth but no isolation

- “We demonstrate that it is possible to impact arbitrary customers’ streams in arbitrary network regions ...” [IMC’08]
Build Efficient Flow Patterns

• High performing P2P content distribution systems build effective flow patterns

• The flow patterns depend on application types and can be the key “secret sauce” of different designers

• We use P2P Live Streaming as an example
P2P Live Streaming Foundation

• Assume that each peer $u$ allocates capacity $C_{uv}$ to a connected neighbor $v$
  • We call $C_{uv}$ the link capacity of the link $u$ to $v$
• Constraints that $\{C_{uv}\}$ should satisfy:
  • Quota: sum of $C_{uv}$ over all neighbors $\{v\}$ of $u$ should be less than the upload capacity of $u$
  • Flow Pattern: For any peer $p$, the maximum flow (minimum cut) from source $s$ to destination peer $p$, under link capacity constraints, should be at least the streaming rate $R$
Live Streaming Feasibility Theorem

• If for every (destination) peer p, the maximum flow computed without other destination peers can support streaming rate $R$, then the streaming system is feasible.
Key insights from the foundation
- It is fundamental that we allocate connectivity and BW to edge capacities for P2P Live Streaming in the correct way
- There are many design options and algorithms to achieve the design

Examples
- Flash crowd acceleration [Wang et al. ‘10]
- Enterprise coordination [Liu et al. ‘10]
- Minimizing server injection points [Alimi et al. ‘10]
Why Related with DECADE?

• A DECADE server conducts resource control just as a peer
  – Controls connectivity to other entities (e.g., peers and/or DECADE servers
  – Uploads to others
  – Downloads from others
  – Manages storage/disk BW

• It is important that DECADE design provides scalable, fundamental “knobs and dials”.