Echelon: Peer-to-Peer Network Diagnosis with Network Coding

IEEE IWQoS 2006
Presented by Chung-Shih Tang
Outline

● Introduction
● Echelon Protocol
● Refining Echelon
● Evaluation
● Conclusion
Introduction (1/2)

- It is critical for operators to monitor performance and “health” of live P2P sessions
  - For P2P applications such as bulk content distribution (e.g. BitTorrent) and live media streaming (e.g. IPTV)
  - Parameters to be measured are application specific
  - These parameters are measured periodically
  - The set of measurements in one time interval is referred to as a snapshot of the peer
  - For long-running P2P applications, most observations are not time sensitive in nature
Introduction (2/2)

- Collecting snapshots
  - **One specific requirement**: the ability to collect snapshots from peers that no longer exist at the time of collection (e.g. left the session or failed)
  - **Traditional wisdom**: rely on peers sending periodic reports to a *logging server*
    - *Not a scalable* design
    - *Remedies*: either decreasing the frequency of obtaining snapshots, or reducing the amount of data to be reported in each snapshot
  - **Primary design objectives of Echelon**
    - Be able to scale to large-scale P2P sessions
    - Tolerate extreme levels of peer dynamics
Echelon Protocol

- **Definitions**
  - $k$ out of $n$ peers periodically collect local snapshots
  - Time interval between two successive snapshots is referred to as an *epoch*, with a length $T$
  - The peers that produce periodic snapshots are called *snapshot peers*, and forms a set $S$
  - There exists a *snapshot collector*, $C$
  - Assume every peer caches coded blocks for $E$ epochs

- **Data message format**

<table>
<thead>
<tr>
<th>Epoch #</th>
<th>ID1</th>
<th>C1</th>
<th>ID2</th>
<th>C2</th>
<th>...</th>
<th>IDk'</th>
<th>Ck'</th>
<th>Coded Data Block</th>
</tr>
</thead>
</table>
Echelon Protocol

- Echelon: an iterative network coding approach
  - Randomized network coding at each peer is further divided into multiple *time slots* of length $t << T$
  - In each time slot, a peer codes from its cached blocks received in the previous time slots, and sends generated blocks to its neighbor peers

- Two remarks about Echelon protocol
  - The iterative protocol execution at each peer does not need to be carefully synchronized
  - Echelon provides excellent resilience to peer dynamics in collecting the network diagnosis
Coded Dissemination

- **At the beginning** of an epoch
  - Collect local measurements & generates an snapshot
  - Each snapshot peer sends its original snapshot to its neighbors
- In each of the **following time slots** $t = 2, 3, \ldots$, a pull-based coded dissemination mechanism is employed based on block advertisement
  - Step 1 – **Advertise** new learned block IDs
  - Step 2 – **Request** to the neighbor with new blocks
  - Step 3 – **Code** and **Deliver** from cached blocks
  - Step 4 – **Cache** the received block if cache not full, otherwise, code received block with a block in cache
Coded Dissemination: An Example

- Four *snapshot peers*: S1, S2, S3, S4
- Each peer can cache up to 4 coded blocks per epoch

At time 3t
Refining Echelon

- Refining the advertising step: to reduce the coded data traffic in the network
  - **Step 1**: peer $i$ sends advertisement messages to randomly selected $NumNeighbor$ neighbors
  - The refined protocol executed at a peer stops when $MaxRound$ rounds has been reached

- Refining the encoding step: to reduce the coefficient overhead in the coding data messages
  - **Step 2**: peer $j$ sends a request containing IDs of the original blocks that it is seeking from peer $i$
  - **Step 3**: peer $i$ generates a new coded block from those containing the original blocks that peer $j$ is seeking
Evaluations

- **Performance metrics**
  - **Rounds**: the maximum number of time slots the iterative protocol is executed at each peer
  - **Decoding Efficiency**: the average number of coded blocks needed to obtain k x k full-rank coefficient matrix for decoding
  - **Number of Peers to Probe**: the average number of peers the snapshot collector has to probe to obtain k coded blocks with
  - **Message Intensity**: the average number of messages sent by each peer in each time slot
  - **Coefficient Overhead**: average size of coefficient part (coefficients & original block IDs) in a data message
Dissemination Speed

- Number of rounds the baseline protocol executes:
  - The protocol stops within $O(\log n)$
  - The protocol terminates faster when peers have more neighbors
Failure Tolerance (1/2)

- Linear independence of resulting cached blocks: any randomly selected $k$ or slightly more than $k$ coded blocks can be used for successful decoding.
Fixed cache capacity = 100

Number of peers to probe is \( \frac{k}{\text{cache capacity}} \), when the cache capacity is small
Message Overhead

- Number of coded data messages is much smaller than that of advertisement messages, especially for larger $d$.
- The coefficient overhead drops a lot when peers have more neighbors.
Comparison with uncoded random dissemination
Effectiveness of Advertising Refinement (1/3)

- The more peers each original blocks is distributed onto in coded form, the better failure tolerance the resulting system has.
Effectiveness of Advertising Refinement (2/3)

- Failure tolerance quickly improves with the increase of $\text{NumNeighbor}$
Effectiveness of Advertising Refinement (3/3)

- Messaging overhead is significantly reduced when the peers are not advertising to all their neighbors.
Effectiveness of Encoding Refinement

- Increased number of probe peers
- Much less coefficient overhead
Conclusion

- *Echelon*, a light-weighted protocol to disseminate peer snapshots over the entire network with network coding, is proposed.
- Utilizing randomized network coding, the dissemination enjoys significant advantages of being bandwidth efficient, scalable and extremely failure tolerant.
- Ongoing work: implementation of *Echelon*.
Discussion

- Issues not addressed in this paper
  - How to choose the $k$ snapshot peers from all $n$ peers in a given network topology
  - How the snapshot collector utilize the snapshots?
- Performance metrics to be further investigated
  - Message overhead arisen from snapshot collection compared to P2P application itself
  - The influence of cache capacity on message overhead and computational overhead
- Apply network coding to time-sensitive applications?
### Linear Network Coding

<table>
<thead>
<tr>
<th>Epoch #</th>
<th>ID1</th>
<th>C1</th>
<th>ID2</th>
<th>C2</th>
<th>...</th>
<th>IDk'</th>
<th>Ck'</th>
<th>Coded Data Block</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Ex:**

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>1</th>
<th>4</th>
<th>4</th>
<th>2</th>
<th>5</th>
<th>7</th>
<th>8</th>
<th>1</th>
<th>Coded Data Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st original data block</td>
<td></td>
<td>X 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4th original data block</td>
<td></td>
<td></td>
<td>X 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5th original data block</td>
<td></td>
<td></td>
<td></td>
<td>X 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8th original data block</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{1st original data block} \times 4 + \text{4th original data block} \times 2 + \text{5th original data block} \times 7 + \text{8th original data block} \times 1 = \text{Coded Data Block} \]