An Adaptive Multipath Routing Algorithm for Maximizing Flow Throughputs

March 6, 2012
Yusuke Shinohara, Yasunobu Chiba and Hideyuki Shimonishi
System Platforms Research Laboratories
NEC Corporation
Outline

- Background
  - Original multipath routing algorithm
  - Problem statement
- Our proposal
- Evaluation results
- Conclusion and future work
Changes in traffic pattern in data center
- Server to server “horizontal traffic”
- Daily and hourly demand change

Solution:
Mesh network + multipath load-balance
Back ground –cont’d

- Difficulty in load balance by ECMP
  - Uses equal cost path ONLY
    - Resolve paths by using packet header and hash function
  - Unawareness of link utilization

- Poor scalability of K-Shortest Path
  - Need huge path computation cost: O(n^3 x N)
    - Depend on # of path candidates

- Key issue:
  Lightweight and efficient multipath selection considering link utilization
Multinomial Logit Based (MLB) routing (1/2)

Multinomial Logit Based Routing

- Based on logit model using random utility theory
- Random path selection based on the path cost of all considerable path
  - E2E path selection based on logit model
  - Enhance hop-by-hop path selection by using equivalent Markov model
- Periodical transition probability computation (O(n³/3))
Multinomial Logit Based (MLB) routing (2/2)

Transition probability \( p( j | i ) \) from node \( i \) to node \( j \) for destination \( d \)

\[
p(i | j) = \exp[-\gamma \cdot c_{ij}] \cdot \frac{W_{jd}}{W_{id}} \]

\( W = [I - A]^{-1} \)

\[
a_{ij} = \begin{cases} 
\exp[-\gamma \cdot c_{ij}] & \text{(If link } e_{ij} \text{ exist)} \\
0 & \text{(Other)} 
\end{cases}
\]

Link cost: Link utilization, delay or loss rate...

Computations

- Matrix \( W \): \( O(n^3/3) \) (periodical)
- Transition probability \( p( j | i ) \): a multiplication and division (at transition)
Problem statement

**Difficulty to determine path diffusion parameter in MLB routing**

- Best value changes dynamically
  - The best value depends on topology and traffic pattern
- Its performance depends on the parameter
  - Large $\gamma$: Tends to select shortest paths and may cause congestion
  - Small $\gamma$: Tends to select various paths and may select unnecessarily detour paths

Dynamic parameter tuning leads to lightweight and effective the multipath routing algorithm.
Overview of the proposed method

- Periodically update the path diffusion parameter
- Search optimum value to minimize the total path cost
  - Path cost: the sum of link utilization that traffic would experience
  - Compute traffic distribution with $\gamma$ and link utilization
  - Estimate future link utilization with traffic distribution and traffic matrix
  - Estimate future path cost with traffic distribution and future link utilization
Parameter tuning with estimation (1/3)

\[ p_{odij} : \text{Probability that the link } e_{ij} \text{ is used by a flow from source } o \text{ to destination } d \]

\[
p_{odij} = \frac{W_{oi} \cdot W_{id}}{W_{od}} \cdot p(j|i) = W_{oi} \cdot \exp[-\gamma \cdot c_{ij}] \cdot W_{jd} / W_{od}
\]
Parameter tuning with estimation (2/3)

- \( l_{ij} \): Traffic amount on link \( e_{ij} \)

\[
l_{ij} = \sum_{o} \sum_{d} (p_{odi} \cdot T_{od})
\]

\( T_{od} \): Traffic matrix from \( o \) to \( d \)

- \( c'_{ij} \): Estimated future Link utilization at the next update

\[
c'_{ij} = c_{ij} + l_{ij} \cdot I_{\gamma} \]

\( b_{w_{ij}} \): Bandwidth of link \( e_{ij} \)

\( I_{\gamma} \): Update interval of \( \gamma \)
Parameter tuning with estimation (3/3)

\( \overline{C}_{od} \): Estimated future average path cost from source \( o \) to destination \( d \)

\[
\overline{C}_{od} = \sum_{i} \sum_{j} p_{odij} \cdot c'_{ij}
\]

Select \( \gamma \) that has the lowest average cost

\( \gamma_n \): current value of \( \gamma \)

\( \gamma_u \): \( \gamma_n \cdot (1 + x) \)

\( \gamma_l \): \( \gamma_n \cdot (1 - x) \)
Experimental evaluation on OpenFlow network

- **OpenFlow Controller**
  - We developed our method on our OpenFlow controller
  - Our OpenFlow controller notify topology to our method
  - Our method create flow entry after resolving path

- **Edge switches**
  - Open vSwitch

- **Core switches**
  - Our prototype OpenFlow switch (48 x GbE + 2 x 10GbE)

- **Servers**
  - Virtual Machine (KVM)
Experimental Evaluation (2/2)

Performance metric
- Link utilization
- Average throughput
- The number of hops
- Parameter $\gamma$

Topology
- Enhanced Hypercube
  - Core switch connects to switch whose hamming distances is one and two
  - Servers: 64, Edge switches: 16, Core switches: 16

Scenario
- Each server has 8 sending thread
- Each sending thread selects destination server randomly and sends for random time (1 – 10 sec) using TCP

Comparison method
- Shortest Path First (SPF)
- MLB Routing with static parameter ($\gamma = 32, 64$)
Our method achieves the best performance by load balancing.
Result (2/2)

Our method tunes the parameter to appropriate value

MLB with static parameter increases the number of hops by selecting redundant paths
Conclusions and Future works

Conclusions
- Needs for mesh network and dynamic routing
- Existing schemes and problem statement
  - MLB routing has difficulty to setup parameter
- Proposal method
  - Dynamic parameter tuning for MLB routing
- Experimental evaluation
  - Our method enhance routing performance by tuning parameter

Future works
- Comparison with other routing algorithms
- Reduction of overhead