Overview

- What is game server discovery?
- Current discovery process
- Previously proposed discovery process[1]
- Further improvements
- Conclusion

Game server discovery

- Generating a list of available game servers
- Players use a ‘game browser’ to find a suitable server in the list to play on
  - ‘Phase 1’
    - Query master server for list of addresses of game servers
  - ‘Phase 2’
    - Query game servers returned by master server sequentially

What’s wrong with the current process?

- Slow
  - Nearly 4 minutes to complete on ‘DSL>256k’ connection
- Resource intensive
  - ~5MB network traffic generated to effectively join a single server
- ‘Brute force’
  - Probes all servers regardless of potential latency to player
Illustration of the issue

- Japan (left) is **distant** to many low RTT servers
- UK (right) is **close** to many low RTT servers
- RTT fluctuates across entire discovery period
- Previously proposed solution aims to
  - Probe low RTT servers before high RTT servers (Algorithm 1)
  - Provide automatic early termination for a given RTT threshold (Algorithm 2)

Algorithm 1: ‘Re-ordering’

1. List of servers
2. Cluster servers by AS
3. Calibration: probing
4. Calibration: splitting
5. Re-ordering
6. Optimised probing
Algorithm 2: ‘Auto-stop’

When ‘bottom’ > maximum tolerated latency (RTT\textsubscript{stop}), stop discovery process.

Why? Re-ordering not perfect.

Illustration of Algorithms 1 & 2

Illustrates previously proposed discovery algorithms (benchmark).

Further improvements?
- Alternative sub-clustering algorithm
- Alternative choices in the number of calibration probes
Alternative sub-clustering algorithm

■ Existing
  □ Split into /16 subnets if the variance within an AS cluster is large

■ Proposed
  □ Split using dynamic network prefix choice if the variance within an AS cluster is large

Algorithm: ‘Alternative sub-clustering’

IP addresses (binary tree)
Illustration of alternative sub-clustering

- Japanese client using existing (left) and alternative (right) sub-clustering algorithms
- Effect by itself on discovery time/traffic appears marginal

Alternative number of calibration probes

- Existing
  - $N_{sample} = \sqrt{N_{cluster}}$
  - $N_{sample} =$ Number of calibration probes
  - $N_{cluster} =$ Number of servers in the AS cluster
- Proposed
  - Scaling $N_{cluster}$
  - Prioritised sampling
Method 1: Scaling $N_{\text{cluster}}$

<table>
<thead>
<tr>
<th>$N_{\text{sample}}$</th>
<th>Calibration probes</th>
<th>Autostop (time and % worst case)</th>
<th>% all probes $&lt;\text{RTT}_{\text{stop}}$ found</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(N_{\text{cluster}})^{0.5}$</td>
<td>3185</td>
<td>74.5s</td>
<td>100.0%</td>
</tr>
<tr>
<td>$(N_{\text{cluster}}/2)^{0.5}$</td>
<td>2391</td>
<td>70.9s</td>
<td>99.9%</td>
</tr>
<tr>
<td>$(N_{\text{cluster}}/4)^{0.5}$</td>
<td>1908</td>
<td>68.3s</td>
<td>99.7%</td>
</tr>
<tr>
<td>$(N_{\text{cluster}}/8)^{0.5}$</td>
<td>1623</td>
<td>66.1s</td>
<td>99.6%</td>
</tr>
</tbody>
</table>

- Negligible trade-off in ‘playable’ servers found for relatively large reductions in time (and hence network traffic too)

Method 2: Prioritised sampling

- Concentrate calibration resources on larger, possibly more diverse AS clusters $\rightarrow$ Use $N_{\text{sample}}$ probes
- Servers within smaller AS clusters less likely to exhibit large variations $\rightarrow$ Use single probe

<table>
<thead>
<tr>
<th>$N_{\text{sample}}$</th>
<th>Calibration probes</th>
<th>Autostop (time and % worst case)</th>
<th>% all probes $&lt;\text{RTT}_{\text{stop}}$ found</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(N_{\text{cluster}})^{0.5}$</td>
<td>1911</td>
<td>68.6s</td>
<td>99.7%</td>
</tr>
<tr>
<td>$(N_{\text{cluster}}/2)^{0.5}$</td>
<td>1680</td>
<td>67.1s</td>
<td>99.7%</td>
</tr>
<tr>
<td>$(N_{\text{cluster}}/4)^{0.5}$</td>
<td>1524</td>
<td>66.1s</td>
<td>99.7%</td>
</tr>
<tr>
<td>$(N_{\text{cluster}}/8)^{0.5}$</td>
<td>1413</td>
<td>65.3s</td>
<td>99.6%</td>
</tr>
</tbody>
</table>
Issue with using fewer calibration probes

- Japanese client using original (left) choice of $N_{\text{sample}}$ and using prioritised sampling (right)
- Undesired effect of poorer re-ordering (not obvious from previous tables)

Combined optimisations

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Calibration probes</th>
<th>Autostop (time and % worst case)</th>
<th>% all probes $&lt;RTT_{\text{stop}}$ found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unmodified</td>
<td>3185</td>
<td>74.5s 32.1%</td>
<td>100%</td>
</tr>
<tr>
<td>Modified</td>
<td>1362</td>
<td>65.1s 28.1%</td>
<td>99.6%</td>
</tr>
<tr>
<td>1 probe per AS</td>
<td>1176</td>
<td>55.8s 24.0%</td>
<td>84.8%</td>
</tr>
</tbody>
</table>

- ‘Modified’ combines both alternative sub-clustering and alternative choice of $N_{\text{sample}}$
  - 43% of original calibration probes (15% more than 1 per AS)
  - 13% less time than original
  - Negligible loss in ‘playable’ servers found
Illustration of combined optimisations

- Japanese client using original ‘unmodified’ algorithm (left) and ‘modified’ algorithm (right)
- Reasonable re-ordering fidelity

Conclusions

- Alternative choice in sub-clustering can improve re-ordering of servers
  - Marginal effects on time/traffic for discovery process
- Alternative choice in $N_{\text{sample}}$ can reduce time/traffic for discovery process
  - Undesired effect of poorer re-ordering
- Alternatives complement each other’s weaknesses
  - Appreciable reductions in time/traffic at negligible cost to accuracy
Thanks

- Grenville Armitage
- Jason But
- Philip Branch