Heterogeneity and Load Balance in Distributed Hash Tables

Brighten Godfrey

Joint Work with Ion Stoica
Computer Science Division, UC Berkeley

IEEE INFOCOM
March 15, 2005

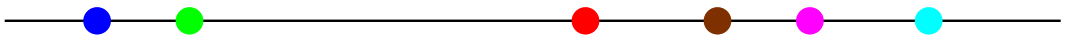
The goals

- Distributed Hash Tables partition an ID space among n nodes
  - Typically: each node picks one random ID
  - Node owns region between its predecessor and its own ID
  - Some nodes get $\log n$ times their fair share of ID space
- **Goal 1:** Fair partitioning of ID space
  - If load distributed uniformly in ID space, then this produces a load balanced system
  - Handle case of heterogeneous node capacities
- **Goal 2:** Use heterogeneity to our advantage to reduce route length in overlay that connects nodes

The goals

- Distributed Hash Tables partition an ID space among n nodes
  - Typically: each node picks one random ID
  - Node owns region between its predecessor and its own ID
  - Some nodes get $\log n$ times their fair share of ID space
- **Goal 1:** Fair partitioning of ID space
  - If load distributed uniformly in ID space, then this produces a load balanced system
  - Handle case of heterogeneous node capacities
- **Goal 2:** Use heterogeneity to our advantage to reduce route length in overlay that connects nodes

The goals

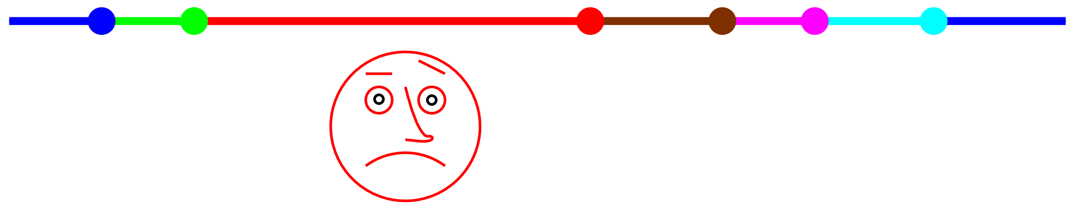


- Distributed Hash Tables partition an ID space among n nodes
  - Typically: each node picks one random ID
  - Node owns region between its predecessor and its own ID
  - Some nodes get $\log n$ times their fair share of ID space
- **Goal 1:** Fair partitioning of ID space
  - If load distributed uniformly in ID space, then this produces a load balanced system
  - Handle case of heterogeneous node capacities
- **Goal 2:** Use heterogeneity to our advantage to reduce route length in overlay that connects nodes

The goals

- Distributed Hash Tables partition an ID space among n nodes
  - Typically: each node picks one random ID
  - Node owns region between its predecessor and its own ID
  - Some nodes get $\log n$ times their fair share of ID space
- **Goal 1:** Fair partitioning of ID space
  - If load distributed uniformly in ID space, then this produces a load balanced system
  - Handle case of heterogeneous node capacities
- **Goal 2:** Use heterogeneity to our advantage to reduce route length in overlay that connects nodes

The goals



- Distributed Hash Tables partition an ID space among n nodes
  - Typically: each node picks one random ID
  - Node owns region between its predecessor and its own ID
  - Some nodes get $\log n$ times their fair share of ID space
- **Goal 1:** Fair partitioning of ID space
  - If load distributed uniformly in ID space, then this produces a load balanced system
  - Handle case of heterogeneous node capacities
- **Goal 2:** Use heterogeneity to our advantage to reduce route length in overlay that connects nodes

Model & performance metric

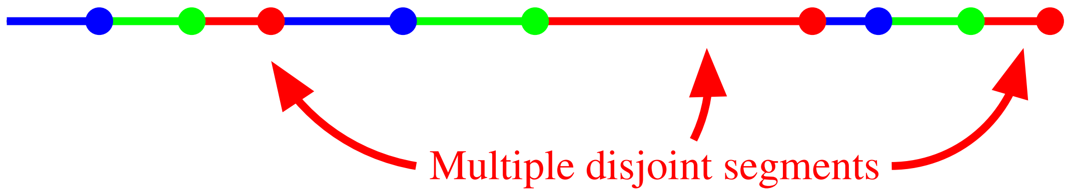
- n nodes
- Each node v has a capacity c_v (e.g. bandwidth)
- Average capacity is 1, total capacity n
- *Share* of node v is

$$\text{share}(v) = \frac{\text{fraction of ID space that } v \text{ owns}}{c_v/n}.$$

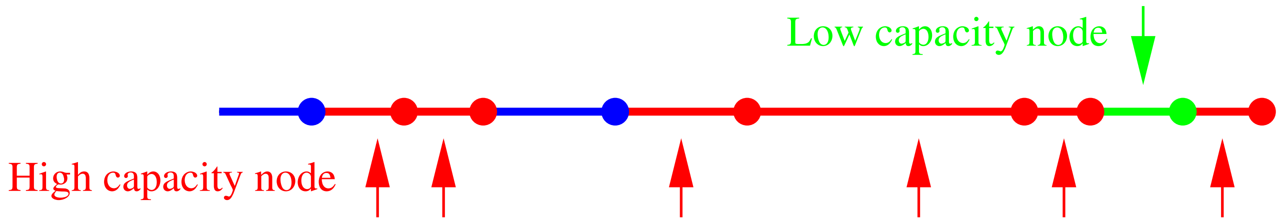
- Want low maximum share
- Perfect partitioning has max. share = 1.

Basic Virtual Server Selection

- **Standard homogeneous case**
  - Each node picks $\Theta(\log n)$ IDs (like simulating $\Theta(\log n)$ nodes)
  - Maximum share is $O(1)$ with high probability (w.h.p.) in homogeneous system



- **Heterogeneous case**
  - Node v simulates $\Theta(c_v \log n)$ nodes (discard low-capacity nodes)
  - Maximum share is $O(1)$ w.h.p. for any capacity distribution



Basic-VSS: Problems

- To route between nodes, construct an *overlay network*
- With $\Theta(\log n)$ IDs, must maintain $\Theta(\log n)$ times as many overlay connections!

- Other proposals use one ID per node, but...
  - all require reassignment of IDs in response to churn, and load movement is costly
  - none handles heterogeneity directly
  - some can't compute node IDs as hash of IP address for security
  - some are limited in the achievable quality of load balance
  - some are complicated

Low Cost Virtual Server Selection

- Pick $\Theta(c_v \log n)$ IDs for node of capacity c_v as before...
- ...but *cluster them* in a random fraction $\Theta(\frac{c_v \log n}{n})$ of the ID space
  - Random starting location r
  - Pick $\Theta(c_v \log n)$ IDs spaced at intervals of $\approx \frac{1}{n}$ (with random perturbation)

- Ownership of ID space is still in disjoint segments
- Why does this help?

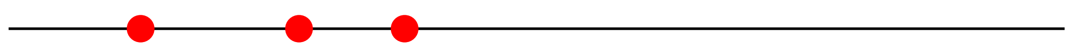
Low Cost Virtual Server Selection

- Pick $\Theta(c_v \log n)$ IDs for node of capacity c_v as before...
- ...but *cluster them* in a random fraction $\Theta(\frac{c_v \log n}{n})$ of the ID space
  - Random starting location r
  - Pick $\Theta(c_v \log n)$ IDs spaced at intervals of $\approx \frac{1}{n}$ (with random perturbation)

- Ownership of ID space is still in disjoint segments
- Why does this help?

Low Cost Virtual Server Selection

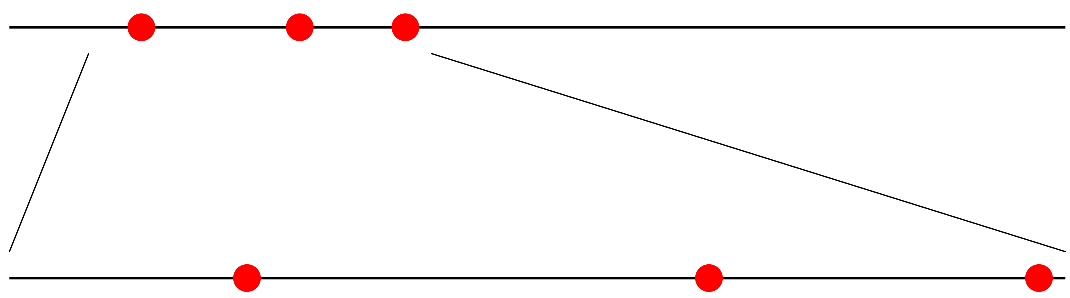
- Pick $\Theta(c_v \log n)$ IDs for node of capacity c_v as before...
- ...but *cluster them* in a random fraction $\Theta(\frac{c_v \log n}{n})$ of the ID space
  - Random starting location r
  - Pick $\Theta(c_v \log n)$ IDs spaced at intervals of $\approx \frac{1}{n}$ (with random perturbation)



- Ownership of ID space is still in disjoint segments
- Why does this help?

Low Cost Virtual Server Selection

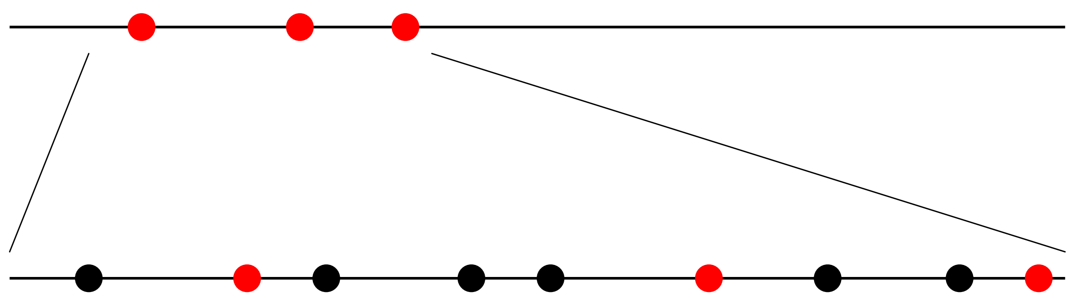
- Pick $\Theta(c_v \log n)$ IDs for node of capacity c_v as before...
- ...but *cluster them* in a random fraction $\Theta(\frac{c_v \log n}{n})$ of the ID space
  - Random starting location r
  - Pick $\Theta(c_v \log n)$ IDs spaced at intervals of $\approx \frac{1}{n}$ (with random perturbation)



- Ownership of ID space is still in disjoint segments
- Why does this help?

Low Cost Virtual Server Selection

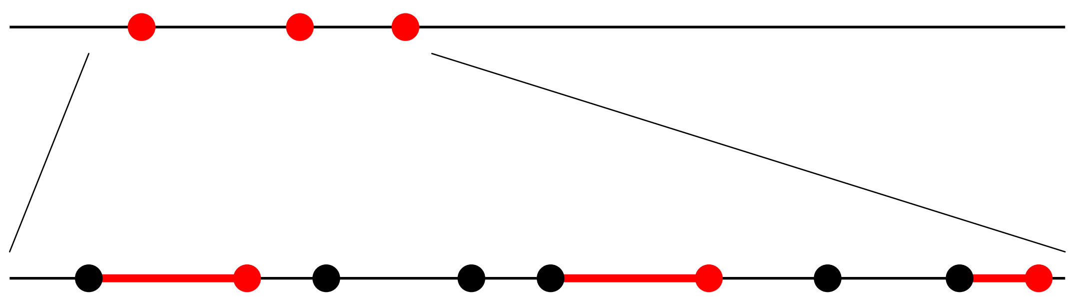
- Pick $\Theta(c_v \log n)$ IDs for node of capacity c_v as before...
- ...but *cluster them* in a random fraction $\Theta(\frac{c_v \log n}{n})$ of the ID space
  - Random starting location r
  - Pick $\Theta(c_v \log n)$ IDs spaced at intervals of $\approx \frac{1}{n}$ (with random perturbation)



- Ownership of ID space is still in disjoint segments
- Why does this help?

Low Cost Virtual Server Selection

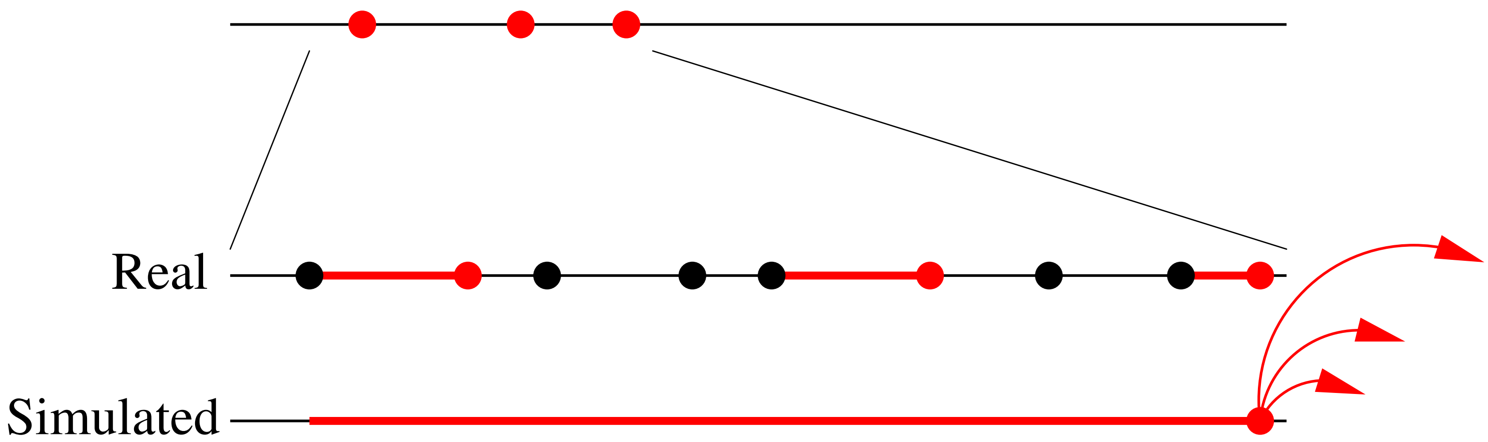
- Pick $\Theta(c_v \log n)$ IDs for node of capacity c_v as before...
- ...but *cluster them* in a random fraction $\Theta(\frac{c_v \log n}{n})$ of the ID space
  - Random starting location r
  - Pick $\Theta(c_v \log n)$ IDs spaced at intervals of $\approx \frac{1}{n}$ (with random perturbation)



- Ownership of ID space is still in disjoint segments
- Why does this help?

LC-VSS: Overlay Topology

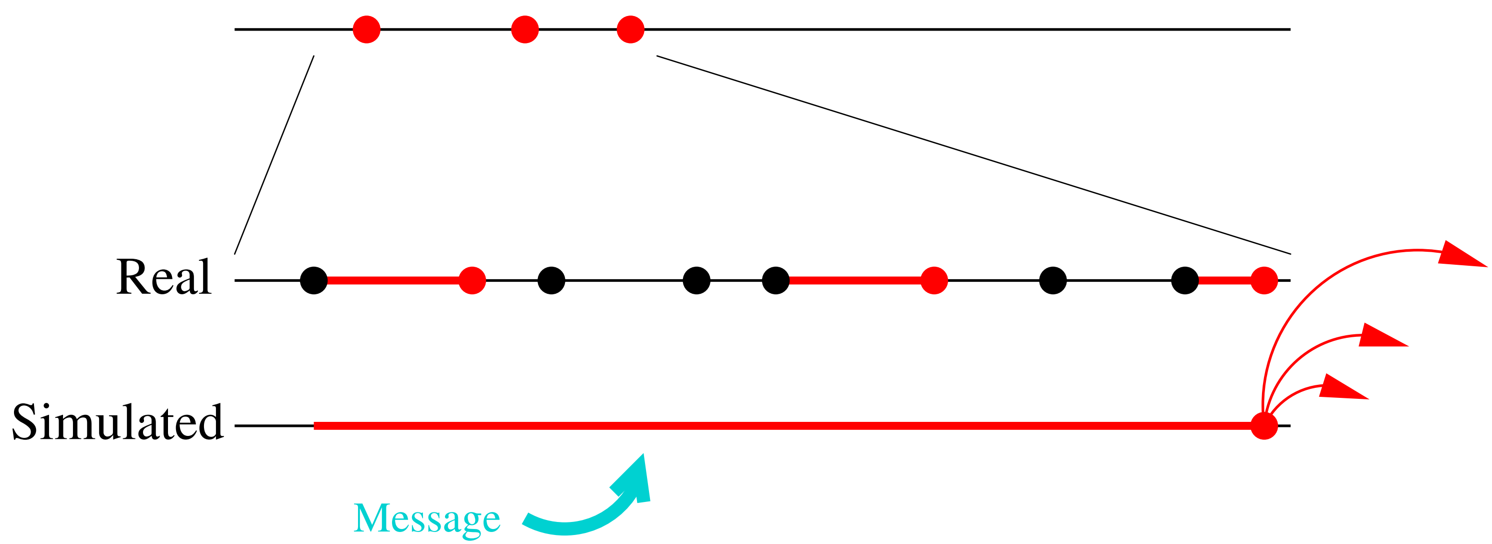
- When building overlay network, *simulate ownership of contiguous fraction* $\Theta(\frac{c_v \log n}{n})$ of ID space



- Routing ends at node *simulating* ownership of target ID, not real owner
- But clustering of IDs $\Rightarrow$ real owner is nearby in ID space $\Rightarrow$ can complete route in $O(1)$ more hops using successor links

LC-VSS: Overlay Topology

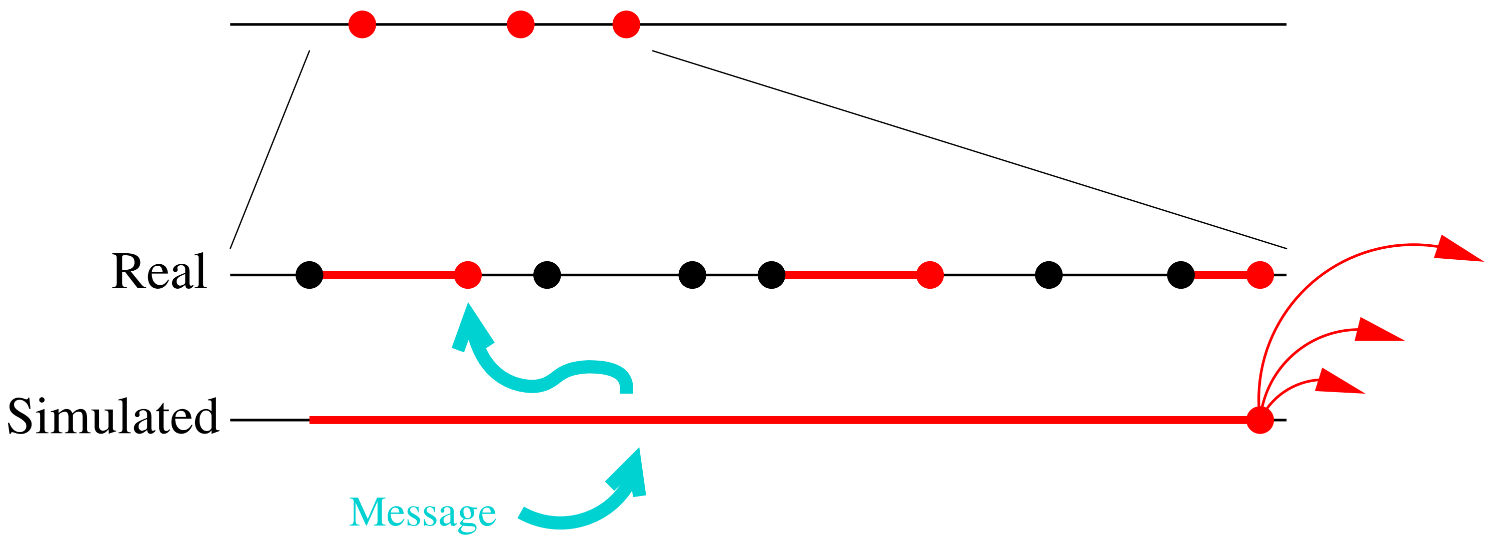
- When building overlay network, *simulate ownership of contiguous fraction* $\Theta(\frac{c_v \log n}{n})$ of ID space



- Routing ends at node *simulating* ownership of target ID, not real owner
- But clustering of IDs $\Rightarrow$ real owner is nearby in ID space $\Rightarrow$ can complete route in $O(1)$ more hops using successor links

LC-VSS: Overlay Topology

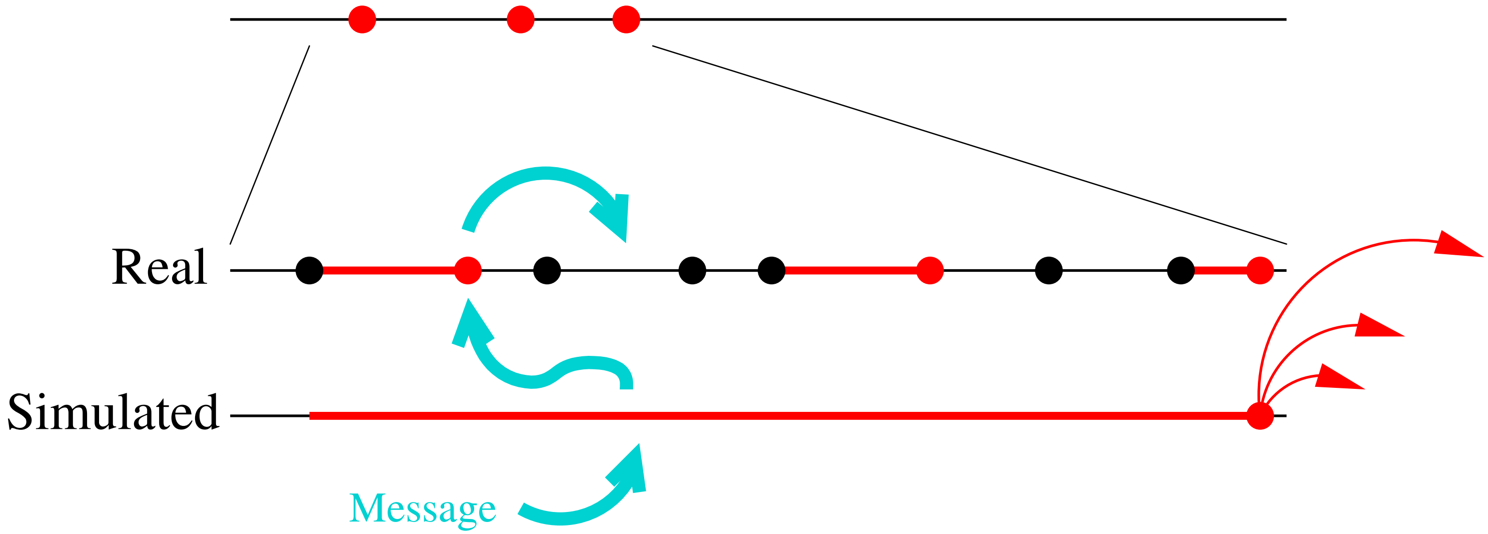
- When building overlay network, *simulate ownership of contiguous fraction* $\Theta(\frac{c_v \log n}{n})$ of ID space



- Routing ends at node *simulating* ownership of target ID, not real owner
- But clustering of IDs $\Rightarrow$ real owner is nearby in ID space $\Rightarrow$ can complete route in $O(1)$ more hops using successor links

LC-VSS: Overlay Topology

- When building overlay network, *simulate ownership of contiguous fraction* $\Theta(\frac{c_v \log n}{n})$ of ID space



- Routing ends at node *simulating* ownership of target ID, not real owner
- But clustering of IDs $\Rightarrow$ real owner is nearby in ID space $\Rightarrow$ can complete route in $O(1)$ more hops using successor links

LC-VSS: Theoretical Properties

- Works for *any* ring-based overlay topology
  - Y_0: LC-VSS applied to Chord
- Compared to single-ID case,
  - Node outdegree increases by at most a constant factor
  - Route length increases by at most an additive constant
- **Goal 1**: Load balance
  - Achieves maximum share of $1 + \varepsilon$ for any $\varepsilon > 0$ and any capacity distribution
    - ...under some assumptions: sufficiently good approximation of n and average capacity, and sufficiently low capacity threshold below which nodes are discarded
  - Tradeoff: outdegree depends on ε

Max Share Proof

Lemma 1 *If node v has at least one ID in the ring and $\alpha = \Theta(\log n)$, then (1) v has between $\alpha c_v/(\gamma_c\gamma_u) - O(1)$ and $\alpha c_v \gamma_c \gamma_u + O(1)$ IDs w.h.p., and (2) v has at least $\gamma_d \alpha(n) - O(1)$ IDs w.h.p.*

Proof: (1) Note that, due to the estimaton error parameters, the factor γ_c lazy update of $\tilde{c}_v$, and the factor 2 lazy update of $\tilde{n}$, we always have $\tilde{c}_v$ within a factor $\gamma_c\gamma_u$ of c_v and $\tilde{n}$ within a factor $2\gamma_n$ of n w.h.p. Thus, for some constant k, the number of IDs that v chooses is at most $\lfloor 0.5 + \tilde{c}_v \alpha(\tilde{n}) \rfloor \leq \tilde{c}_v \alpha(\tilde{n}) + O(1) \leq \gamma_c \gamma_u c_v k \log(2\gamma_n n) + O(1) \leq \gamma_c\gamma_u \alpha(n) + O(1)$. The lower bound follows similarly, noting that we are not concerned with nodes that have been discarded. (2) Similarly, if v has decided to stay in the ring, we must have $\tilde{c}_v \geq \gamma_d$ and the bound follows by the above technique. ■

We now break the ring into *frames* of length equal to the smallest spacing parameter s_{min} used by any node. The following lemma implies that $s_{min} \geq 1/(2\gamma_n n)$ w.h.p.

Lemma 2 *Let $\beta = (1 - \gamma_c\gamma_u\gamma_d)/(\gamma_c\gamma_u)$. When $\alpha \geq \frac{8\gamma_n}{\beta\varepsilon^2} \ln n$, each frame contains at least $(1-\varepsilon)\beta\alpha n s_{min} - O(1)$ IDs w.h.p. for any $\varepsilon > 0$.*

Proof: Assume that no node has more than one ID in any frame; if this is not the case, we can break the high-capacity nodes for which it is false into multiple "virtual nodes" without disturbing the rest of the proof.

Consider any particular frame f. Let X_v be the indicator variable for the event that node v chooses an ID in f and let $X = \sum_v X$. We wish to lower-bound X. Suppose v chooses m_v points. Since f covers a fraction s_{min} of the ID space, we have $E[X_v] = m_v s_{min}$. By Lemma 1, $m_v \geq \alpha c_v/(\gamma_c\gamma_u) - O(1)$ for nodes R in the ring. Thus,

$$\begin{aligned}
\mathrm{E}[X] &= \sum_{v\in R} \mathrm{E}[X_v] \\
&\geq \sum_{v\in R} s_{min}\left(\alpha c_v/(\gamma_c\gamma_u) - O(1)\right) \quad \text{(Lemma 1)} \\
&\geq -O(1) + \sum_{v\in R} \frac{s_{min}\alpha c_v}{\gamma_c\gamma_u} \\
&= -O(1) + \frac{s_{min}\alpha}{\gamma_c\gamma_u}\sum_{v\in R} c_v \\
&\geq -O(1) + \frac{s_{min}\alpha}{\gamma_c\gamma_u}\cdot(1-\gamma_c\gamma_u\gamma_d)n \quad \text{(Claim ??)} \\
&= \beta\alpha n s_{min} - O(1),
\end{aligned}$$

with β defined as in the lemma statement. (Note that although Claim **??** was stated in the context of Chord, it applies to our partitioning scheme without modification.) A Chernoff bound tells us that

$$\begin{aligned}
\Pr[X < (1-\varepsilon)\mathrm{E}[X]] &< e^{-(\beta\alpha n s_{min} - O(1))\varepsilon^2/2} \\
&= O(e^{-\beta\alpha n s_{min}\varepsilon^2/2}) \\
&< e^{-\beta\alpha\varepsilon^2/(4\gamma_n)} \quad \text{(Lemma ??)} \\
&= O(n^{-2})
\end{aligned}$$

when $\alpha \geq \frac{8\gamma_n}{\beta\varepsilon^2}\ln n$. Again by Lemma **??**, there are at $\leq 2\gamma_d n$ frames, so the lemma follows from a union bound over them. ■

Proof: (Of Theorem **??**) If node v is discarded, its share is 0, so we need only consider nodes in the ring. Such a node v chooses one ID in each of $m \leq \alpha c_v \gamma_c\gamma_u + O(1)$ frames (Lemma 1).

We first fix the nodes' choices of the frames in which they place their IDs. Let $X_1, \ldots, X_m$ be the fraction of the ID space owned by each of node v's IDs. The randomness in the X_is is over the intra-frame positions of the nodes' IDs, which are chosen independently and uniformly at random. By Lemma 2, we may assume that each frame has at least one ID. Thus, the interval assigned to the ith ID may span at most one frame boundary, so X_i depends only on the locations of the IDs in its frame and in the counterclockwise preceding frame. Thus, the odd-indexed X_is are mutually independent, as are the even-indexed X_is. We will bound the share of these two groups in the same way, one at a time. Consider first the odd-indexed X_is.

Break each frame into d *buckets* of equal size; we'll pick d later. A bucket is *occupied* when some node other than v chooses an ID inside it, and is *empty* otherwise. To analyze the node v's share of the ID space, we'll count the number of empty buckets counterclockwise-following v's chosen IDs. Define an infinite sequence of random variables Y_j, each of which will be the indicator variable for the event that a particular bucket is occupied. Y_1 will correspond to the bucket counterclockwise-following v's first odd-indexed ID. Suppose Y_j corresponds to the kth bucket following v's ℓth ID. Then we have two cases. (1) If $Y_j = 0$, Y_{j+1} corresponds to the next bucket following the same ID. (2) Otherwise, Y_{j+1} corresponds to the first bucket following the next odd-indexed ID, i.e. the $(\ell+2)$th one. If $m/2 < \ell + 2$ then we simply set $Y_{j+1} = 1$. Thus, the number of zeros in the sequence of Y_j's is the number of buckets entirely owned by v's $m/2$ odd-indexed IDs.

With the goal of upper-bounding the number of zeros, we first deal with dependence among the Y_js. By Lemma 2 we may assume that each frame has at least $r = (1-\varepsilon)\beta s_{min} n\alpha(n) - O(1)$ IDs for sufficiently large α. View $Y_1, Y_2, \ldots$ as a process. If $Y_{j-1} = 1$, then we are in Case (2) and Y_j corresponds to a frame independent of those of $Y_1, \ldots, Y_{j-1}$, so there are at least r IDs distributed u.a.r. in the frame which may occupy Y_j's bucket. If we are in Case (1) then Y_j's bucket is in the same frame as that of Y_{j-1}, which implies that some of the buckets in that frame are empty, in which case there are at least r IDs distributed u.a.r. in a *subset* of the frame including Y_j's bucket. This discussion implies that, regardless of the history of the Y_js, the probability that $Y_j = 1$ is at least $1 - (1 - 1/d)^r$. Formally, we define another sequence of variables Z_j which are independent Poisson trials with success probability p to be picked below. For any indeces $j_1, \ldots, j_k$, we have

$$\begin{aligned}
\Pr[Y_{j_1} = \cdots = Y_{j_k} = 1] &= \prod_{\ell=1}^{k} \Pr[Y_{j_\ell} = 1 | Y_{j_1} = \cdots = Y_{j_{\ell-1}} = 1] \\
&\geq \prod_{\ell=1}^{k}\left(1 - \left(1 - \frac{1}{d}\right)^r\right) \\
&\geq \left(1 - e^{-r/d}\right)^k \\
&= \Pr[Z_{j_1} = \cdots = Z_{j_k} = 1]
\end{aligned}$$

where we have chosen the success probability for the Z_js to be $p = 1 - e^{-r/d}$. This implies that an upper bound the number of 0's in the independent Z_j sequence is also an upper bound the number of 0's in the dependent Y_k sequence, a fact which we use next.

If we see $m/2$ ones in the first x Y_js, then by the definition of the sequence, we have seen all the zeros, of which there are at most $x - m/2$. Thus node v will own at most $x - m/2$ complete buckets, plus $2 \cdot m/2$ partial buckets (one at each end of the $m/2$ contiguous sequences of complete buckets), for a total of at most $x + m/2$ buckets due to its $m/2$ odd-numbered IDs. We now show that we see the required $m/2$ succeses w.h.p. when $x = \frac{m}{2p(1-\delta)}$. Let P be the number of 1's in the first x Y_js, and let P' be the corresponding value for the Z_js. By the above discussion we have $\Pr[P < m/2] \leq \Pr[P' < m/2]$, and $\mathrm{E}[P'] = xp = \frac{m}{2(1-\delta)}$, so

$$\begin{aligned}
\Pr[P < m/2] &\leq \Pr[P' < m/2] \\
&= \Pr[P' < (1-\delta)\cdot\frac{m}{2(1-\delta)}] \\
&\leq e^{-\frac{m\delta^2}{4(1-\delta)}} \quad \text{(Chernoff bound)} \\
&\leq O(e^{-\frac{\gamma_d\alpha\delta^2}{4(1-\delta)}}) \quad \text{(Lemma 1 part (2))} \\
&= O(n^{-2})
\end{aligned}$$

when $\alpha \geq \frac{8(1-\delta)}{\gamma_d\delta^2}\ln n$. In this case, counting now both odd- and even-indexed points, node v owns at most $m + \frac{m}{p(1-\delta)}$ buckets, each of size s_{min}/d. Normalizing by v's fair share c_v/n, we have

$$\mathrm{share}(v) \leq \frac{1}{c_v/n}\cdot\left(\frac{m s_{min}}{d} + \frac{m s_{min}}{dp(1-\delta)}\right).$$

Recall that d is arbitrary. Taking the limit as $d \to \infty$, we have $dp \to r = (1-\varepsilon)\beta s_{min} n\alpha(n) - O(1)$ so

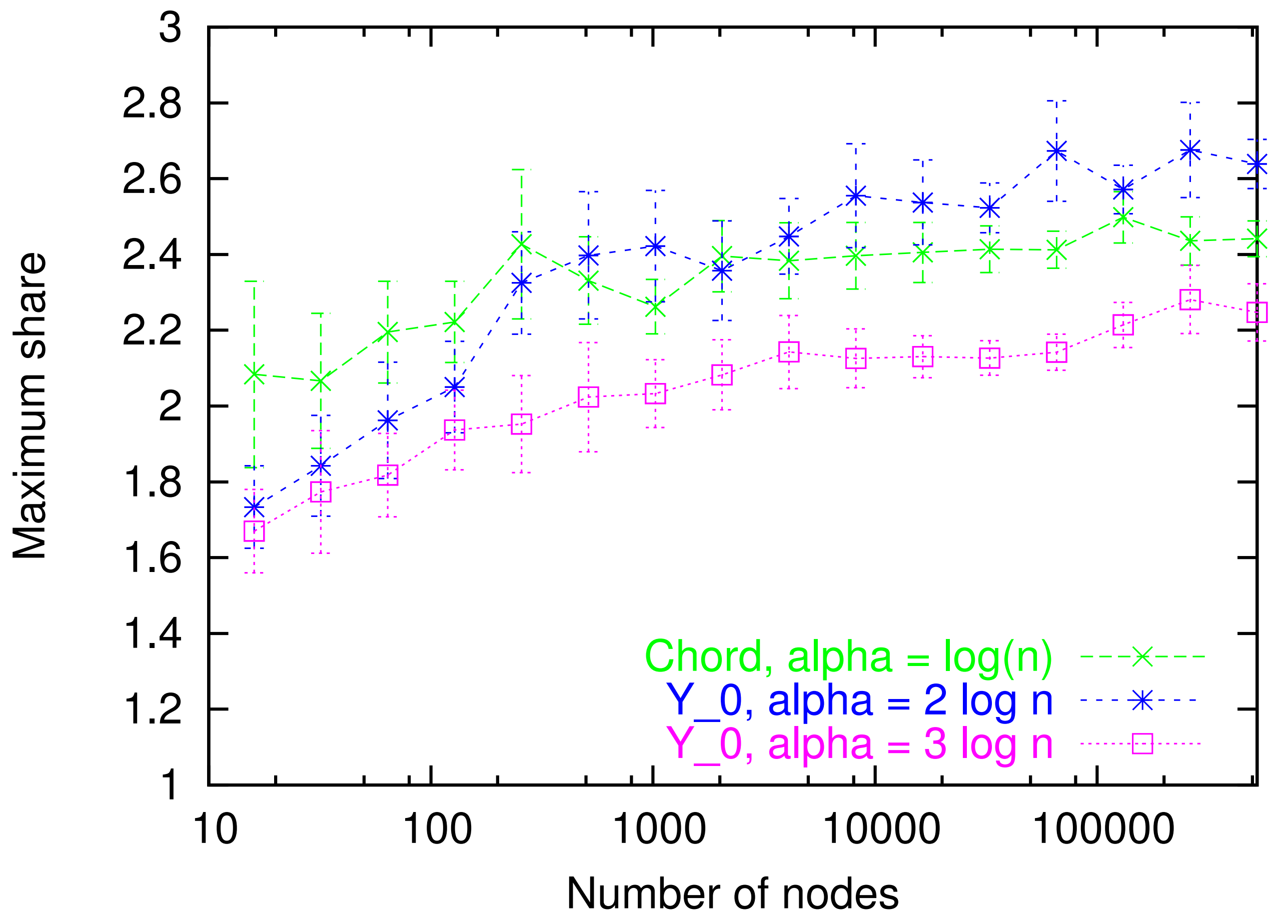
$$\begin{aligned}
\mathrm{share}(v) &\leq \frac{1}{c_v/n}\cdot\frac{m s_{min}}{(1-\delta)((1-\varepsilon)\beta s_{min} n\alpha(n) - O(1))} \\
&\leq \frac{1}{c_v}\cdot\frac{m}{(1-\delta)(1-\varepsilon)(1-\varepsilon')\beta\alpha(n)} \\
&\leq \frac{1}{c_v}\cdot\frac{\alpha(n)c_v\gamma_c\gamma_u + O(1)}{(1-\delta)(1-\varepsilon)(1-\varepsilon')\beta\alpha(n)} \quad \text{(Lemma 1 part (1))} \\
&\leq \frac{(1+\varepsilon'')(\gamma_c\gamma_u)^2}{(1-\delta)(1-\varepsilon)(1-\gamma_c\gamma_u\gamma_d)}
\end{aligned}$$

with probability $1 - O(n^{-2})$ for any $\varepsilon', \varepsilon'' > 0$ and sufficiently large n, so by a union bound, this is true of all nodes w.h.p. Finally, we require that α is the maximum of the requirement given above and that of Lemma 2; setting $\delta = \varepsilon$ for convenience of presentation, we have $\max\{\frac{8(1-\varepsilon)\ln n}{\gamma_d\varepsilon^2}, \frac{8\gamma_n\gamma_c\gamma_u\ln n}{(1-\gamma_c\gamma_u\gamma_d)\varepsilon^2}\} \leq \frac{8\gamma_n\gamma_c\gamma_u\ln n}{(1-\gamma_c\gamma_u\gamma_d)\gamma_d\varepsilon^2}$, as required by the theorem. ■

Simulation

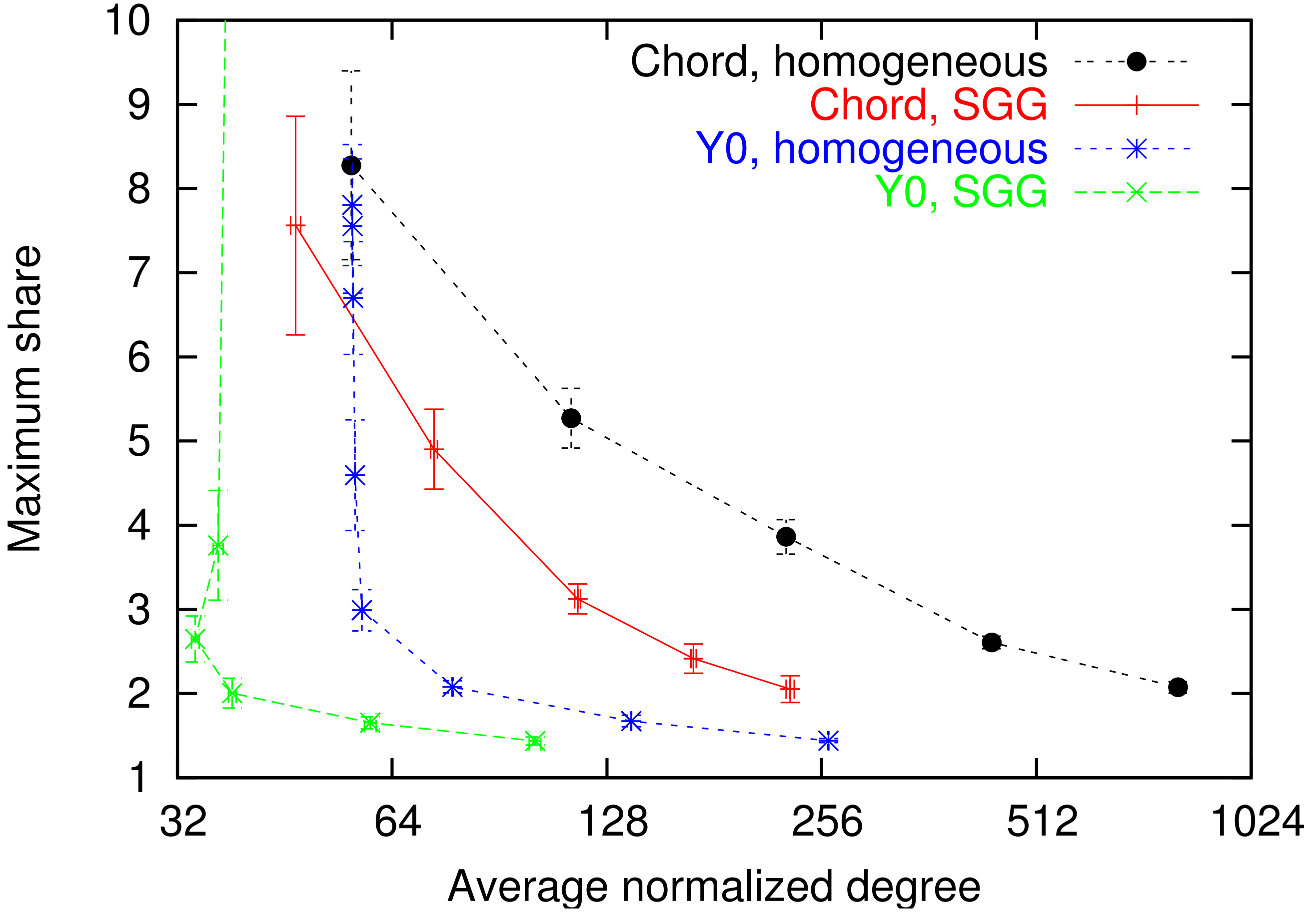
- The Contestants
  - Chord: Basic Virtual Server Selection
  - Y_0: LC-VSS applied to Chord's overlay topology
- Static simulator
  - Important simplification: Nodes know n and average capacity
  - These would actually be estimated
  - and there would be some "lazy update" to provide hysteresis

Simulation: Maximum share



- **Parameter:** α = number of virtual servers per unit capacity
- Homogeneous capacities shown here
- Chord with $\alpha = 1$ increases to maximum share ≈ 13.7.

Max Share/Degree Tradeoff

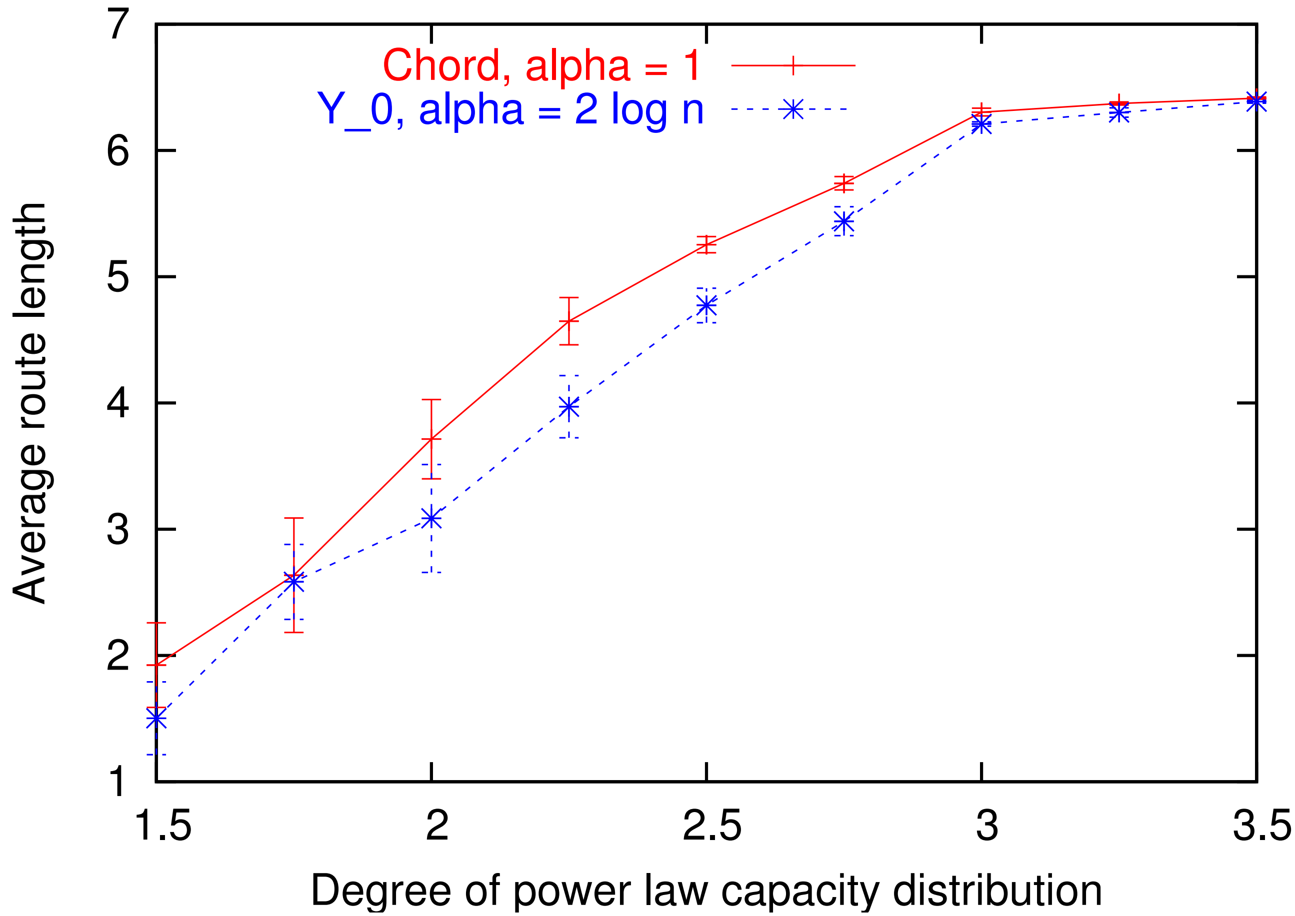


$n = 2048$. Points achieved with $\alpha \in \{1, 2, 4, 8, 16\}$ for Chord, and $\alpha \in \{1, 2, 4, \ldots, 128\}$ for Y_0.

Goal 2: Exploit Heterogeneity

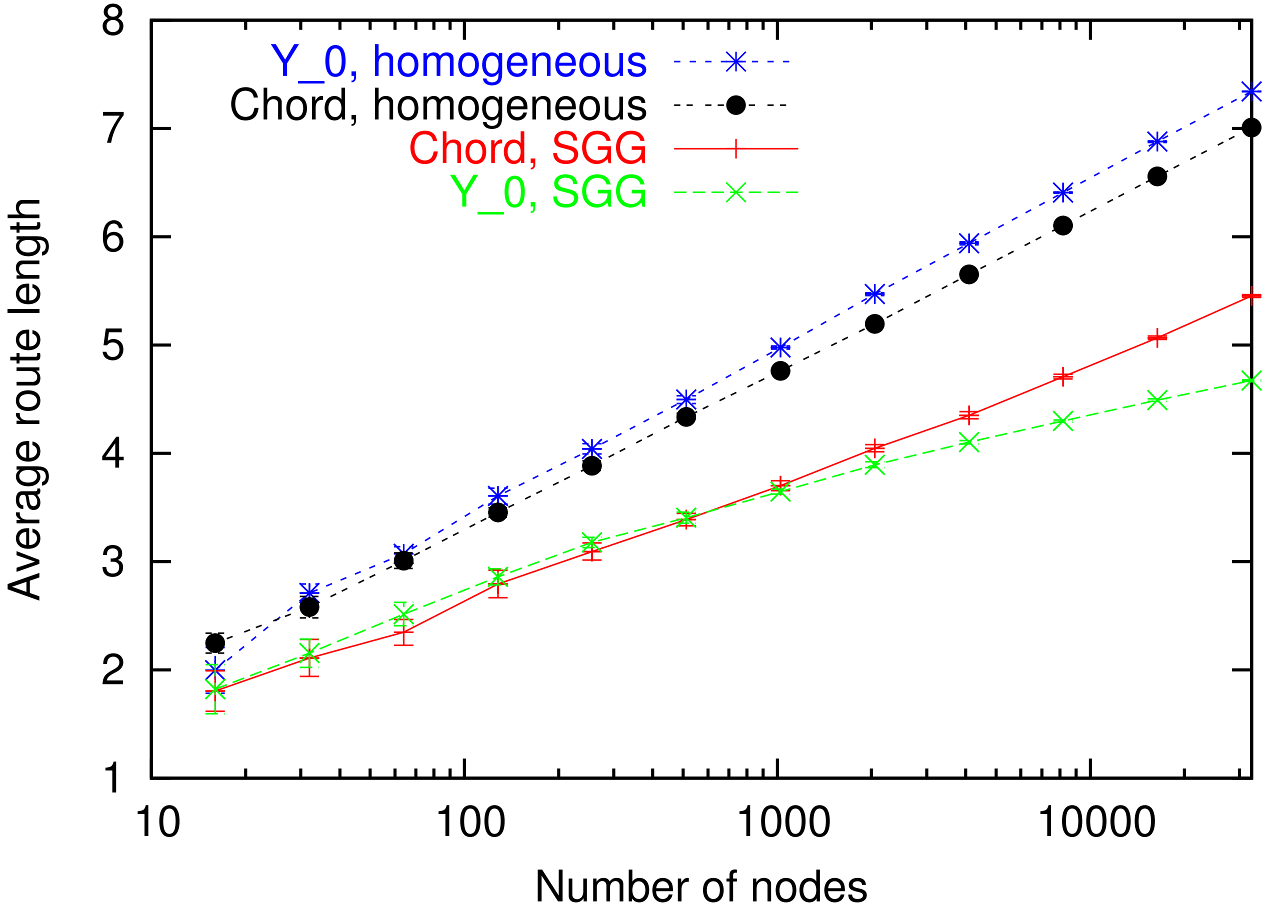
- Even high-capacity nodes have a single set of overlay links
- Make use of unused capacity: pick denser set of links
- In Chord with $\alpha = 1$: $\Theta(c_v \log n)$ total outlinks
  - $\Theta(\log n)$ links in $\Theta(c_v)$ finger tables (one per virtual server)
- In our scheme: $\Theta(c_v \log n)$ total outlinks
  - ... all in one dense finger table
  - More structured topology $\Rightarrow$ reduced route length

Simulation: Effect of heterogeneity



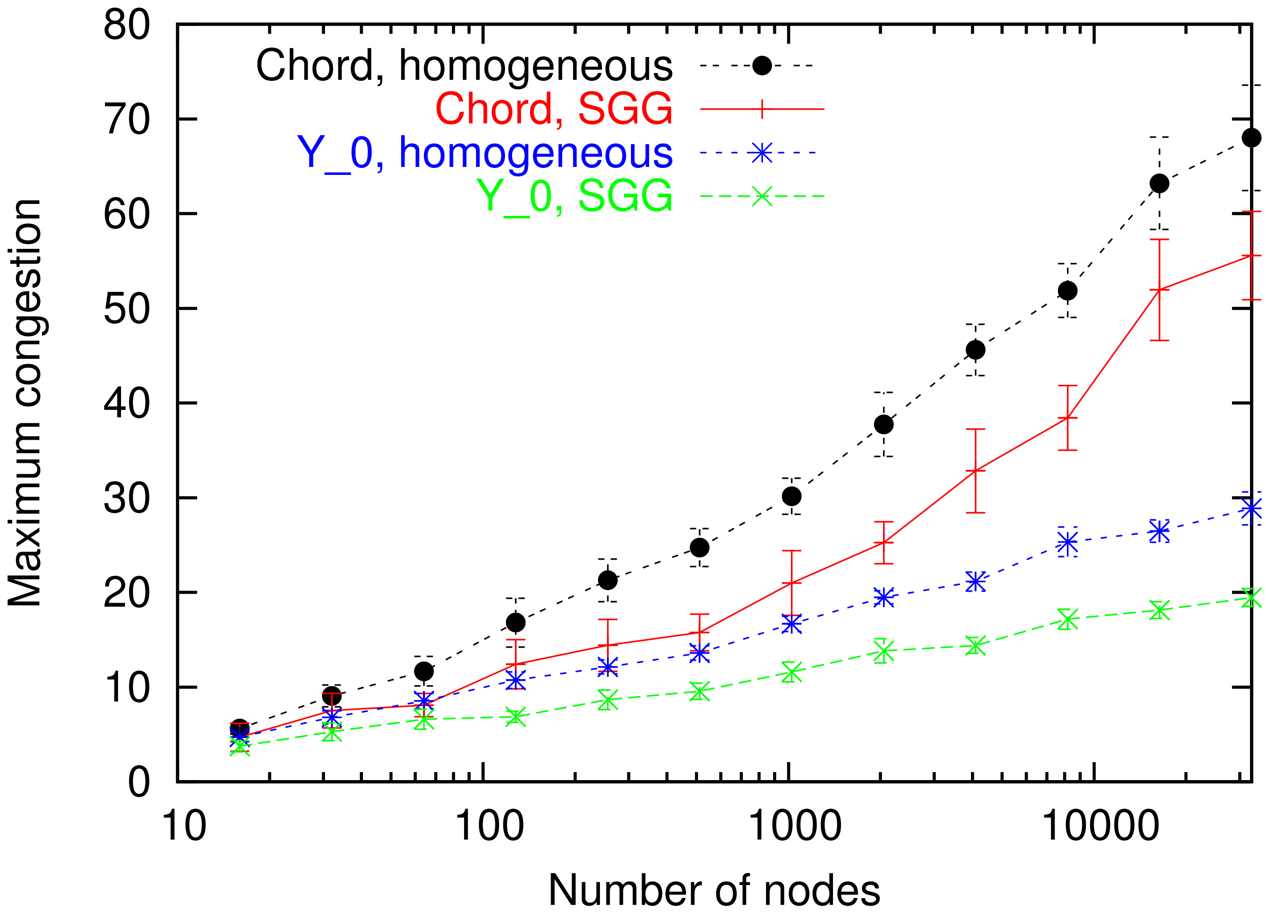
Route length vs. capacity distribution in a 16,384-node system.

Simulation: Effect of heterogeneity



- SGG capacity distribution from real Gnutella hosts
- *Asymptotic* route lengths compared to homogeneous case
  Chord: $\leq$ 23% shorter Y_0: $\geq$ 55% shorter

Simulation: Congestion

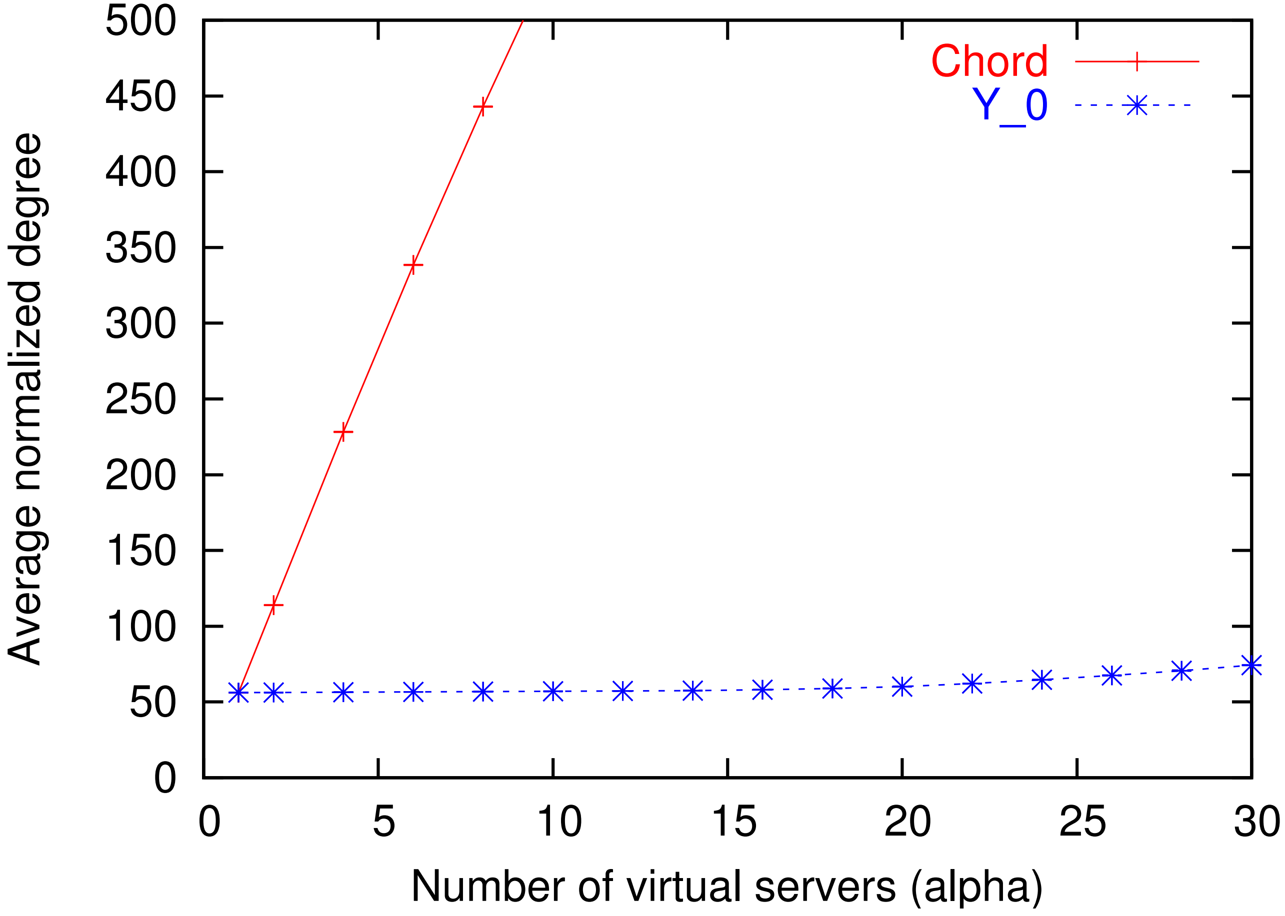


Conclusion

- **Costs**
  - Some additional overhead, especially when particularly good balance desired
  - Will incur additional load movement when number of nodes or average capacity changes by a constant factor
  - Require estimates of n and average capacity
  - Assumes uniform distribution of load in ID space
- **Benefits**
  - Simple way to achieve good load balance at low cost
  - Compatible with any ring-based overlay
  - Adds flexibility in neighbor selection to any overlay
  - Takes advantage of heterogeneity to reduce route length

Backup slides

Simulation: Degree



Degree of a node = number of links to other nodes

Simulation: Max Share vs. Capacity Distribution