Capturing Complexity in Networked Systems Design: The Case for Improved Metrics

Sylvia Ratnasamy
U.C. Berkeley
Motivation

Our community values “simple”, “clean” system designs

“The most important lesson we learned is the value of simple designs…”
   – Google Bigtable, OSDI ’06

“The complexity of the architectural design…” – Sprint, IEEE Network 2000

“A design for multicast that is simple…” – Perlman. Simple Multicast, IETF Draft.

“The advantage of Chord is that it is less complicated…” – Chord, Sigcomm’01

“Complex routing algorithms may have difficulty scaling… simple floods are sufficient” – TinyOS, NSDI’04
Motivation

Our community values “simple”, “clean” system designs

• But lacks metrics that rigorously capture this aesthetic

• Best practice: metrics borrowed from the theory community
  • e.g., total_messages, total_state

• Problem: at times incongruent with our notion of “simplicity”
  • e.g., flooding: inefficient but simple
  • e.g., dijkstra’s: global state, but simple
  • e.g., leaderID vs. neighborID
The NetComplex Work

• Can we quantify design “simplicity”?  
  – more rigorously compare-and-contrast design options  
  – align design goals of algorithms, systems communities

• First stab: complexity metrics for the algorithmic component of a networked system

• Note
  – just one aspect to system complexity
  – intent: complement, not replace, existing metrics
Outline

• Sketch of a solution
• Sample results
• Limitations and future directions
Approach

Builds on two observations:

– much of system design centers around state
  • what state is required?
  • how is it constructed/maintained?
  • how is it used?

– what distinguishes wide-area state:
  • derived from remote nodes → dependencies are distributed
  • relayed via intermediate nodes → more dependencies

Current metrics mostly treat all state as equal
Proposal

For a given piece of state $s$, measure the ensemble of distributed dependencies that yield $s$

• First cut: measure $\rightarrow$ counting
  – akin to existing metrics: #msgs, #state
  – amenable to evaluation by simple examination, simulation
(Obs. 1) Types of Dependencies

s is value dependent on x,
s is transport dependent on next(B) state at A, R1, R2, R3.

\[ v_s = \#\text{pieces of state on which } s \text{ is value dependent} \]
\[ t_{s \leftarrow x} = \#\text{pieces of state relied on to transport } x \text{ to } s \]
(Obs. 1) Types of Dependencies

- $A \xrightarrow{\text{next}(B)} R1 \xrightarrow{\text{next}(B)} R2 \xrightarrow{\text{next}(B)} R3 \xrightarrow{\text{next}(B)} B$

- $x = 30^\circ$

- $v_x = t_x = 0$

- $v_s = 1; t_s = 4$
(Obs. 2) Value Dependencies
Accumulate

\[ y = 20^\circ \]
\[ v_y = t_y \leftarrow y = 0 \]

\[ x = 30 + y^\circ \]
\[ v_x = 1 \]
\[ t_x \leftarrow y = 2 \]

\[ s = x \]
\[ v_s = 2 \]
\[ t_s \leftarrow x = 4 \]
Dependencies $\rightarrow$ Complexity?

How do the number of value and transport dependencies for a piece of state $s$, translate to the complexity of $s$?
Summation isn’t good enough

\[
R1: x = y + b \\
R2: y = z + c \\
R3: z = d
\]

value dependencies = 3
transport dependencies = 6

\[
A: s = x \\
B: x = y + b \\
C: y = z + c \\
D: z = d
\]

value dependencies = 3
transport dependencies = 6
Must capture the propagation structure of dependencies
Final Metric

Complexity of $s$: $c_s = u_{s \leftarrow x} + t_{s \leftarrow x} + c_x$

- $u_{s \leftarrow x} = \text{number of state changes relayed from } x \text{ to } s$
- $t_{s \leftarrow x} = \text{complexity of transport states for relaying from } x \text{ to } s$
- $c_x = \text{complexity of state } x$
Extensions

1. Complexity of state $s$ that is derived from multiple inputs $x_1, x_2, \ldots, x_m$ – distinguish between $s$ derived from ALL, ANY or k-of-m

2. Complexity of a function is derived from the complexity of the state it acts on – e.g., “route(msg)” acts on forwarding state at every hop
Recap

• Compute the complexity of each piece of state $s$
  – if $s$ is local state, $c_s = 0$
  – else identify the inputs $x_1, x_2, \ldots x_m$ that $s$ is derived from
    and calculate $c_s$ from $c_{x_1}, c_{x_2} \ldots c_{x_m}$ and $t_{s\leftarrow x_1} \ldots t_{s\leftarrow x_m}$

• Compute the complexity of a function from the complexity of the state it acts on

• System complexity
  – sum of the complexity of its functions and/or
  – sum of the complexity of its states
Outline

- Sketch of a solution
- Sample results
- Limitations and future directions
Evaluation: Routing

(n: number of nodes, f: node degree, d: network diameter = log n)

<table>
<thead>
<tr>
<th>Routing scheme</th>
<th>Complexity</th>
<th>State</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compact (AG+04)</td>
<td>$O(nd^2)$</td>
<td>$O(\sqrt{n})$</td>
<td>$O(n\sqrt{n})$</td>
</tr>
<tr>
<td>Routing on Flat Labels (data-centric)</td>
<td>$O(d^2/\log^2 n)$</td>
<td>$O(\log n)$</td>
<td>$O(n\log^2 n)$</td>
</tr>
<tr>
<td>Distance Vector</td>
<td>$O(d^3)$</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Link State</td>
<td>$O(d^2)$</td>
<td>$O(nf)$</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>Centralized (a la SDN)</td>
<td>$O(d^2)$</td>
<td>$O(nf)$; $O(n)$</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>Centralized + Source Routing</td>
<td>$O(d)$</td>
<td>$O(nf)$; $O(n)$</td>
<td>$O(n^3)$</td>
</tr>
</tbody>
</table>
**Evaluation: classical distributed systems**

<table>
<thead>
<tr>
<th>Category</th>
<th>System</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared read/write variable</td>
<td>Quorum</td>
<td>$O(k^2)$</td>
</tr>
<tr>
<td></td>
<td>Read one/write all available</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Stronger consistency has higher complexity</td>
<td>Two phase commit</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Stronger fault tolerance has higher complexity</td>
<td>Gossip</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Efficiency need not be congruent with complexity</td>
<td>TTL-based</td>
<td>$\varepsilon$</td>
</tr>
<tr>
<td>Soft-state is viewed as simpler than hard-state</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(n: number of nodes, k: quorum size)

- Stronger consistency has higher complexity
- Stronger fault tolerance has higher complexity
- Efficiency need not be congruent with complexity
- Soft-state is viewed as simpler than hard-state
Open Questions

• Extending netcomplex to capture:
  – correctness/quality
  – robustness
  – correlated inputs

• Beyond algorithmic design complexity
  – configuration complexity
  – software complexity
  – complexity in the service model
Summary

• Design simplicity: valued, but poorly measured

• Our question: is design complexity measurable?

• Conjecture: capturing dependencies in state is key (w.r.t. algorithmic component of a system)

• Open: metrics that capture other aspects to system complexity (configuration, code, service model)
Thanks! Questions?
Complexity of operations

- An operation acts on different pieces of state.
- Each piece of state has a complexity.
- Compute complexity of an operation by (appropriately) combining complexity of state it acts on.

The complexity of forwarding operation: $c_{\text{fwd}} = \mathcal{O}(d^3)$

Graph with states $A$, $s=0$, $s=1$, $s=2$, ..., $s=i$, $s=d$, $B$.

- $c_s = 0$, $1$, ..., $\mathcal{O}((d-1)^2)$, $\mathcal{O}(d^2)$.

$x=0$ $s=1$ $s=2$ $s=i$ $s=d$
complexity of operations

- an operation acts on different pieces of state
- each piece of state has a complexity
- compute complexity of an operation by (appropriately) combining complexity of state it acts on

complexity of fetch: $c_{\text{fetch}} = O(k/m)$