The National Institute of Standards & Technology (NIST) is a measurement laboratory within the U.S. Government. Founded in 1901, NIST is a non-regulatory federal agency within the Department of Commerce. NIST's mission is to promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve quality of life. NIST is a multidisciplinary organization operating across many scientific domains, including: physics, chemistry, materials and information technology. I work within NIST’s Information Technology Laboratory, which also directs the well-known NIST Cloud Computing and Cyber Security programs.

To illustrate my ideas in this talk, I identify a number of companies offering cloud computing services. While those references encompass a broad cross-section of the industry, I do not identify every cloud-computing company in the market. The fact that I identify certain cloud-computing companies does not imply endorsement by NIST, nor does such identification suggest that those companies are the best available.
There is an undeniable trend away from corporate-owned data centers and toward adoption of cloud-computing services. The trend appears no matter how it is measured: (1) number of objects stored in the cloud, (2) number of cloud customers or (3) dollars spent on cloud computing. This trend means that our information-based economy exhibits growing dependence on large data centers deployed and managed by a handful of information technology companies, and accessible from the Internet. This trend implies that enterprises are moving from paying capital + operating expenses to just paying operating expenses. On the other hand, failures within a handful of large cloud-computing infrastructures can have widespread effects, when compared with failures distributed across a large number of corporate data centers.
Over the past six years, many cloud outages and performance degradations have occurred, and the consequences have been costly. These cloud failures and degradations have been spread widely across the cloud industry rather than being concentrated in one or two cloud vendors. These cloud failures and degradations stem from various root causes, including: power outages due to storms, failures in backup generators, software bugs, configuration errors and insufficient reliability in applications designed for deployment on clouds.

Assuming that the rate of such problems stays constant over the coming decade, the growing reliance on cloud services will increase the consequences of failures and degradations, substantially increasing the cost to our information-based economy. Bear in mind that the companies building and using cloud systems and services are quite technically advanced, and very competent in designing and deploying large, distributed systems. Despite this fact, reliability issues continue to plague large, distributed, complex information systems.
My thesis is that failures and performance degradations in such systems do not arise from any inability to design and deploy large-scale systems and distributed applications, but rather from a knowledge gap. We lack the scientific and engineering knowledge needed to understand, predict and control behavior in complex information systems, such as clouds, the Internet and distributed systems deployed on them. This knowledge gap threatens to undermine society’s growing reliance on the complex information systems that underlie our modern economy.

While there are many reasons why complex information systems are difficult to understand & predict, I am going to mention three: (1) state-space explosion, (2) emergent behaviors and (3) heavy-tailed distributions.
Complex information systems comprise a vast state space. This means that there are many system states that could exist, and finding those that would lead to system-wide failures and performance degradations is quite challenging. Even in an abstract simulation model of a complex information system the state space is immense. For example, NIST has developed Koala, a simulator of an infrastructure-as-a-service (IaaS) cloud. Koala has about 130 parameters that one might wish to explore for failure scenarios and performance degradations. Though each parameter could be represented by a floating-point number, we can reduce the state-space by restricting the range of parameter values to search to a small subset, say 6 values per parameter ($6^{130}$). Even with such drastic restriction, the Koala search space encompasses $>10^{100}$ combinations, which exceeds the number of atoms ($10^{80}$) in the visible universe. Further, simulations can produce myriad responses, some of which are duplicative, others unique and essential. For example, the NIST Koala simulator can exhibit 200 or more responses, which might in reality reflect around a dozen unique behavioral dimensions. When searching the Koala state space, analysts must be sure to measure all the unique behaviors, but not to overweigh overlapping responses.
Many key, global properties of complex information systems appear as emergent behaviors, which can be impossible to predict based on detailed analyses of the behavior of individual system components. Such global properties can only be discovered by measuring macroscopic variables of a system under an appropriate set of conditions that stimulate the spatiotemporal emergence of the behaviors of interest. Changing system components in some way intended to create a particular outcome can result in global emergent behavior that is quite different than expected.

For example, we simulated a distributed grid computing system, where a population of users attempted to complete multitask, workflow jobs. The graph plots (y axis) the probability density function (pdf) of the times (x axis) taken to complete jobs. The black curve represents baseline system performance. Subsequently, we injected a domain-name system (DNS) spoofing attack, where some sites pretended to be willing to execute user tasks, accepted user submissions and then did not execute the jobs. Affected users had to detect such events and then resubmit their jobs to other, real service providers. As a result, the blue curve shows the effects of DNS spoofing on the pdf of job execution times. We then imagined the developer of the grid client released a software update that contained sensible behavior to combat DNS spoofing. Specifically, once an suspected spoofer is detected, the client places the suspect in a penalty box and does not attempt to use the suspect until some penalty period expires. As the red curve shows, the updated client software worsened rather than improved system performance, causing a larger proportion of jobs to be later than before the patch was released. This result occurred because the global schedule of task executions is an emergence property of a grid system. By improving the ability of clients to combat DNS spoofing attacks, more clients became competitive in the same time period for the limited processing resources in the grid, and the schedule of task executions was shifted, causing more work-flows to be delayed in their overall completion times.
In a best-selling book, Nassim Nicholas Taleb dubbed rare events as black swans, which appear infrequently in a world where swans are mainly white. In complex information systems, large-scale system failures can often be attributable to such rare events, which can lead to very costly outcomes. Most traditional analytical modeling of information systems adopts Gaussian or Poissonian assumptions because the underlying probability distributions are amenable to tractable mathematical analyses. In an effort to validate such analytical models, simulations of information systems usually adopt the same assumptions. In such probability distributions, the likelihood of improbable events, > 3 sigma from the mean, is vanishingly small, thus rare, potentially costly, outcomes occur quite infrequently. Taleb argues that Gaussian assumptions do not adequately represent the true probability distribution of rare events in complex systems. Instead, Taleb proposes that heavy-tailed distributions, such as Pareto and log-normal more closely represent reality. In such heavy-tailed distributions, the occurrence of rare events is more highly probable than in a Gaussian distribution. In other words, rare events that can lead to system-wide failures are more likely then assumed in the usual approach to systems modeling. By adopting optimistic assumptions when simulating system behavior, analysts obtain a biased and erroneous view about the potential for unexpected rare events to cause disruptions.

For the past six years, my colleagues and I at NIST have been investigating techniques that can be used to understand & predict behavior in complex information systems. Next I will give a high-level overview of our research. Then, I will focus on one particular research objective that we have just begun pursuing over the past year.
From 2006-2010, my colleagues & I investigated and evaluated methods to understand the influence of distributed control algorithms on global system behavior and user experience. We focused on two case studies: (1) comparing proposed replacements for the TCP congestion-control algorithm and (2) comparing alternative algorithms for allocating virtual machines to physical machines in infrastructure clouds. The work was general, and inherently multidisciplinary, spanning computer science, mathematics, statistics, data mining and information visualization. As you will see, this work was quite successful.

Beginning in 2011, we turned our attention to investigating and evaluating methods to increase reliability in complex information systems. Our focus to date has been on infrastructure clouds, where we could leverage the Koala simulator developed under our previous research, and also on TCP/IP networks, where we could leverage our MesoNet simulator, used in the study of Internet congestion-control algorithms.

The work remains inherently multidisciplinary, and we have a possibility to collaborate with others. If you are interested please let me know, either at the conference or later via email.
The primary focus of our past research was on: (1) state-space reduction methods (for parameters and responses) and (2) measuring global system behaviors.

In the main, our research from 2006 to 2010 focused on methods to determine what to measure under what conditions at an affordable cost when comparing competing distributed control algorithms for complex information systems.

We evaluated a wide range of techniques to predict how network behavior and user experience would change if TCP were replaced by any of seven competing congestion control algorithms.

We then used the most effective of those techniques to compare various algorithms for allocating virtual machines to physical nodes in Infrastructure as a Service clouds.

Next I introduce a few of the techniques that proved effective.
Upper left-hand quadrant of the slide: To reduce the parameter search space of system models, we employed a combination of parameter reduction techniques (including model restriction, factor clustering and 2-level orthogonal fractional factorial experiment design) to reduce the experiment parameter space for a TCP/IP network model from $10^{539}$ to $2^5$. These reductions enabled us to compare TCP with 7 proposed replacements under only 32 conditions, requiring 256 simulations for a given experiment scenario. Such parsimony allowed us to investigate 5 scenarios, with $(8 \times 32 \times 5 =) 1280$ simulations, leading to the most comprehensive study to date of proposed TCP replacements.

Upper right-hand quadrant: To reduce the response space, we used correlation analysis and clustering. The example shows an application of correlation analysis and clustering to reduce 40 responses in the Koala cloud-computing model to an 8-dimensional response space. This means that the model can be characterized by measuring only 8 responses.

Lower left-hand quadrant: An example taken from our study of potential TCP replacements. Here, we do a cluster analysis of all responses for each of seven congestion control algorithms under 32 different parameter combinations. The plots in blue indicate that for uncongested networks, varying congestion-control algorithm makes little difference in response measurements. In fact, for the least congested condition (12) there is no difference in system behavior, regardless of congestion-control algorithm used. For a number of other conditions, congestion control algorithm 3 stands out as distinctly different from the others.

Lower right-hand quadrant: To investigate causality, we resorted to more detailed analysis methods, such as the one shown. Here, we measure residual differences in retransmission rate for each of the seven congestion control algorithms under the 32 conditions. We sort...
the conditions by increasing residual value. Notice that algorithm 3 is an outlier in over half of the conditions, increasing with increasing in congestion. So, here we can see that increased network congestion leads to increased retransmission rates for algorithm 3.
We produced a report on our comparison of proposed TCP replacements: (1) we found that the proposed replacements would not significantly change global network behavior, and we were able to explain why; (2) we identified 5 conditions that must hold for a user to gain advantage from any of the proposed TCP replacements; (3) we identified one of the proposed TCP replacements as most suitable for deployment; and (4) we raised cautionary notes about another proposed TCP replacement. To demonstrate generality, we applied the same methods to compare alternative VM-placement algorithms for on-demand clouds.

We also produced a number of papers, each outlining some of our methods, as applied to either TCP/IP networks or IaaS clouds.

While our state-space reduction methods allow us to characterize global system behaviors under a wide range of parameter combinations, these approaches do not address directly the search for low-probability, highly costly failure scenarios and performance degradations. That is the focus of our ongoing research.
To increase reliability in complex information systems, we are investigating two general classes of methods: (1) design-time methods and (2) run-time methods. We aim for design-time methods that system engineers can use to detect existence and causes of costly failure regimes prior to system deployment and (2) develop run-time methods that system managers can use to detect onset of costly failure regimes in deployed systems, prior to collapse.

We are investigating three design-time methods. First, we hope to exploit state-space reduction techniques from our previous research to narrow the parameter combinations that need to be considered. Second, we combine Markov chains with graph and perturbation analysis to identify and characterize selected system failure modes. We are also investigating the use of genetic algorithms to drive system models into anti-optimal behavioral directions. Third, we are interested in determining whether percolation (i.e., spreading) processes are at work, when information systems are driven toward phase transitions.

Before I turn to a discussion of our ongoing investigation of run-time methods, let me give a brief idea of two of our design-time methods, devoting one slide to each.
Here we construct Markov chain models, with probabilities determined by instrumenting systems (or models) operating under nominal conditions, and then treating those models as graphs, which can be subjected to cut-set analysis to determine subsets of edges that would disconnect the graph if removed. Those subsets represent potential failure scenarios. We then use perturbation analysis on those subsets to determine numeric thresholds at which the system fails, and also collapse trajectories. We have published a number of papers on this approach.
Here, we use a genetic algorithm to drive a population of model simulators from an initially random set of parameter combinations toward parameter combinations that exhibit some anti-optimal behavior. During this process, we collect anti-fitness reports that include parameter-value pairs and associated anti-fitness values. Subsequently, we examine these reports to search of combinations of parameter-value pairs that are associated with anti-optimal system behaviors. We have found this computationally costly technique successful, and useful when schedule permits and the cost of failures is high.
Our initial work on run-time methods is motivated by many reports and demonstrations of phase transitions from uncongested to congested phases, as reported in the physics literature for simulations of abstract network models.

We are currently investigating the four questions listed here. If we find that the demonstrated phase transitions entail a spreading (i.e., percolation) process and occur in realistic network models, then it might be possible to identify precursor patterns and measurement methods that can signal oncoming phase transitions in time to take corrective actions.

While we are investigating these ideas in the context of network congestion, they should also apply in other failure processes that spread over space and time.

We are currently focusing on the first of two of these four research questions.

Next, I will describe what we have found so far.
To begin our investigation, we selected the work of Echenique, Gomez-Gardenes and Moreno (hereafter referred to as EGM), who demonstrated phase transitions from uncongested to congested network regimes. We selected their work because (unlike most of the other papers in physics journals) they based their model on a real network topology, an 11174-node snapshot (circa 2001) from the Internet Autonomous System topology.

On the left, we show a visualization of the EGM topology. On the right we show a log-log plot of the node degree (x axis) vs. the frequency (y axis) for the EGM topology. The plot demonstrates that this topology is a scale free network with a slope of -2.2. Though, the method for deriving the -2.2 slope is somewhat ad hoc.
Based on this topology, EGM simulate a network where packets are injected, forwarded to their destination and then consumed. The EGM simulation assumes that nodes can hold an infinite number of packets. At each time step, $p$ packets are injected, with the injection nodes selected randomly – based on a uniform distribution. Destination nodes are also chosen randomly (uniform distribution) for each packet. Also at each time step, each node forwards the first packet in its queue to a next-hop neighbor node, unless the node is the packet’s destination, in which case the packet is consumed, i.e., leaves the network. If the node is not the packet’s destination, the next hop is chosen as the neighbor with the minimum delta, where delta for each neighbor is the sum of the $h$-weighted distance in hops to the destination and the $(1-h)$weighted queue length of the neighbor. When $h=1$, this amounts to shortest-path routing, as typically used in the Internet. When $h < 1$, this is a traffic aware routing that considers the packet backlog in the neighbors, where $h=0$ means that the next-hop choice is chosen strictly based on congestion.

The system response is measured as rho, the proportion of injected packets that are queued in the network. As rho approaches 1, all injected packets are queued in the network. Higher rho indicates a higher rate of network congestion.
The graph on the left shows the results obtained when EGM simulated their model for $h=1$ and for three values of $h<1$. Note that for $h=1$ (shortest path routing) after the injection rate ($p$) passes 3, the packet backlog in the network begins to increase, almost linearly. This is indicative of a second-order phase transition.

For simulation with $h<1$, the packet backlog does not increase until the injection rate passes 8 or 9. In these cases, the increase jumps discontinuously, somewhat like a first-order phase transition.

The graph on the right shows our replication of the EGM simulation for $h=1$ and for $h=0.85$. Our results seem congruent with the EGM results, thus we infer we have correctly implemented their simulations. We use our simulation to build upon the results of EGM.
The first question we investigate is whether or not similar phase transitions occur in smaller networks.

Here, we use a three-tiered, 218-node, topology based on a US Internet Service Provider.

Our simulations show the same second- and first-order phase-transition behavior, as exhibited for the single-tiered, 111174-node topology used by EGM.

Our next question is whether or not these phase transitions can be explained by a spreading process, i.e., a percolation process?

Before addressing this question, let me give a one-slide tutorial on percolation theory.
Percolation involves the spreading, in space and time, over a network, of some property of interest. The probability that a given node possesses that property is designated by \( p \). In our case, the property of interest is congestion, though the property of interest might be a disease, a virus, a failure, and so on. Since our property of interest is congestion, \( p \), will be the packet injection rate.

In percolation, as a property spreads, we can consider the sub-graphs of connected nodes that possess the property. We can measure the largest of such sub-graphs as a giant connected component (GCC), which is measured as \( P_\infty \) - the proportion of nodes within the GCC. Since our property of interest is congestion, \( P_\infty \) is the proportion of nodes in the largest congested sub-graph. We declare nodes congested if the packet queue size exceeds some arbitrary value – here, we select 5 packets.

In percolation, there exists a critical value \( p_c \) such that when \( p \) exceeds that value, the GCC grows proportionally to \( (p - p_c)^\beta \).

So, when \( p \) is below \( p_c \) no GCC forms, but when as \( p \) exceeds \( p_c \) the GCC grows exponentially.

If the spread of congestion in a network is a percolation process, then we should be able to use this theory to detect the spread of congestion before the entire network is congested. Thus, we could signal a network operator to take some remedial action to forestall the spread of congestion before the entire network is affected.
To determine if spreading congestion is a percolation process, we simulated the three-tiered, 218-node, ISP network as before, but this time measuring the size of the GCC ($P_\infty$).

The top graph measures $P_\infty$ with increasing $p$ for shortest-path routing ($h=1$).

The bottom graph measures $P_\infty$ with increasing $p$ for congestion-aware routing ($h=0.85$).

As you can see, both simulations reveal a percolation process, where the size of the GCC increases ($h=1$) or jumps ($h=0.85$) to just over 20%, where it remains until $p$ passes some critical point, either $p=160$ (for $h=1$) or $p=140$ (for $h=0.85$), after which the size of the GCC increases steeply until the GCC includes all nodes (i.e., $P_\infty = 1$).

From this we infer that: (1) congestion spreading in a network is a percolation process and (2) there may be the prospect of monitoring the increase in the congested GCC to alert a network operator before the size of the congested GCC passes 25% of the nodes in a network.
Here, we introduce one realistic characteristic: limiting queue sizes ($q$), first to 10 packets and then to 100 packets.

In both simulations, we set $h=0.85$.

The graphs show that percolation does occur under limited queues.

The top graph also shows that limiting the queue size to only 10 packets transforms the percolation process for congestion-aware routing into the shape of the percolation process for shortest-path routing under infinite queues.

The bottom graph shows that a larger queue size of 100 packets returns the percolation process to the shape seen for congestion-aware routing under infinite queue size.
The next question for us to investigate surrounds conditions under which the percolation of the network holds.

Does percolation occur under more realistic assumptions?

If so, under what conditions? If not, then why not?
Given affirmative answers to the preceding questions, we would be encouraged to investigate two subsequent questions:

(1) If congestion-based phase transitions do occur in realistic network models, are there precursor patterns of behavior (as appear for abstract network models) that can be used to signal incipient transitions so that network managers can take remedial actions to avoid an undesired transition, for example, from a free-flowing state to a jammed state?

(2) If precursor behavior patterns do exist, are there implementable detection methods that could be inserted into operational networks to provide effective signaling?

As our research continues, stay tuned for our answers to these questions.
Thanks for coming today, and for listening to my remarks.

I wish to credit my collaborators, Chris Dabrowski, Jim Filliben and Sandy Ressler, for their contributions to this research.

I also wish to acknowledge the programmatic leadership provided by Ron Boisvert, the NIST program manager for research in complex systems.

You can explore our research in more detail at the indicated URL.

Any additional questions?