MODELING THE INTERNET

Pamela Zave

AT&T Laboratories—Research

Florham Park, New Jersey, USA

MODELING THE INTERNET: OUTLINE

WHY SHOULD IT BE MODELED?

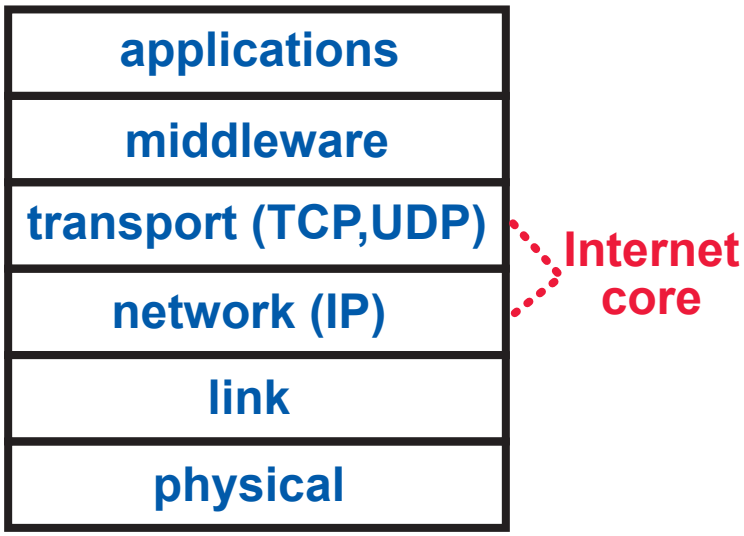
WHAT SHOULD BE MODELED?

HOW SHOULD IT BE MODELED?

STATE OF THE "CLASSIC" INTERNET ARCHITECTURE

THE "CLASSIC" INTERNET ARCHITECTURE

- defined in terms of layers with different functions



- designed to empower users and encourage innovation
- has succeeded beyond anyone's wildest dreams

THE REAL INTERNET

- the classic architecture has been made obsolete by explosive growth in users, traffic, applications, and security threats
- it does not meet current or future needs for . . .
  - . . . security and reliability,
  - . . . quality of service,
  - . . . resource management,
  - . . . balancing the interests of diverse stakeholders such as multiple service providers and their customers

[Clark et al. 05]

- the good properties of the classic architecture are eroded badly by workarounds

THE STATE OF INTERNET APPLICATIONS

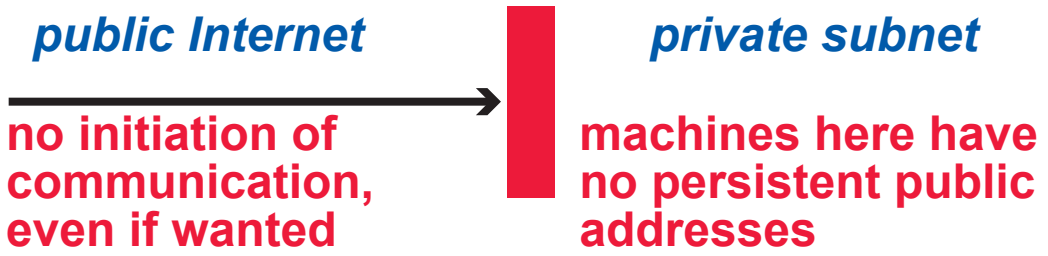
MANY APPLICATION NEEDS ARE NOT SUPPORTED

- machine mobility
- process and virtual-machine migration
- session-level anycast
- new name spaces
- scaling up
- application-level security and privacy
- balancing stakeholder interests

e.g., Web proxies can be invoked on behalf of users, authorities, Internet service providers, content providers

THE COSTS AND BENEFITS OF NETWORK SECURITY

Network-level security is provided by Network Address Translation (NAT) and firewalls.



- applications are not secure!
- yet this form of security makes it very difficult to build peer-to-peer applications
- NAT and firewalls are so tightly inter-twined with TCP and UDP that unless they know about an application, it is unlikely to work

i.e., HTTP

- even applications that work are difficult to build, deploy, and maintain because there is no separation of concerns

THE STATE OF INTERNET EVOLUTION

[Handley 06]

INTERNET "OSSIFICATION"

- there has been no important change in the transport layer (TCP/ UDP) since 1988
- there has been no important change in the network layer (IP) since 1993

" . . . technologies get deployed in the core of the Internet

when they solve an immediate problem

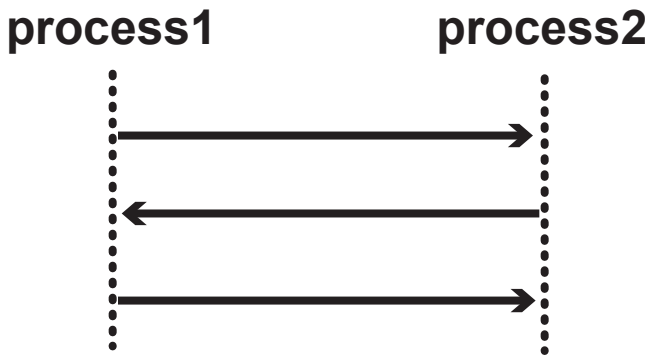
or when money can be made"

- it is very difficult for an Internet service provider to make money with improvements, because most have no effect until everyone else adopts them
- a crisis is the only way to get the global consensus required for real change

THE STATE OF THE IETF . . . AS CAPTURED BY THE SPECIFICATION OF SIP

THE MEDIUM

- IETF philosophy is to standardize based on "rough consensus and working code"
- finite-state machines are rarely used
- specifications are written in English, augmented only by message sequence charts that look like this (IETF macros):



note how this forces you to forget race conditions!

THE MESSAGE

- the base document (IETF RFC 3261) is 268 pages
- it is continually being extended, bottom-up, in response to an endless series of new use cases
- "A Hitchhiker's Guide to SIP" is a snapshot of SIP RFCs and drafts . . .

  . . . which lists **142** documents, totaling many thousands of pages

- opinions are based on two false assumptions:

more generality can only be obtained with more complexity

a bad scenario can be ignored if you claim it is rare (a "corner case")

this situation makes it very difficult to build SIP applications

THE STATE OF NETWORKING RESEARCH

[Rexford 10]

"In my college networking class I fell asleep at the start of the semester when the IP header was on the screen, and woke up at the end of the semester with the TCP header on the screen."

THE STATE OF NETWORKING RESEARCH

[Rexford 10]

"In my college networking class I fell asleep at the start of the semester when the IP header was on the screen, and woke up at the end of the semester with the TCP header on the screen."

"Networking is all details and no principles."

THE STATE OF NETWORKING RESEARCH

[Rexford 10]

"In my college networking class I fell asleep at the start of the semester when the IP header was on the screen, and woke up at the end of the semester with the TCP header on the screen."

"Networking is all details and no principles."

"There is a tendency in our field to believe that everything we currently use is a paragon of engineering, rather than a snapshot of our understanding at the time. We build great myths of spin about how what we have done is the only way to do it . . . to the point that our universities now teach the flaws to students . . . who don't know better." —John Day

THE STATE OF NETWORKING RESEARCH

[Rexford 10]

"In my college networking class I fell asleep at the start of the semester when the IP header was on the screen, and woke up at the end of the semester with the TCP header on the screen."

"Networking is all details and no principles."

"There is a tendency in our field to believe that everything we currently use is a paragon of engineering, rather than a snapshot of our understanding at the time. We build great myths of spin about how what we have done is the only way to do it . . . to the point that our universities now teach the flaws to students . . . who don't know better." —John Day

"So, these network research people today aren't doing theory, and yet they aren't the people who brought us the Internet. What exactly are they doing?"

MODELING THE INTERNET: OUTLINE

WHY SHOULD IT BE MODELED?

Because businesses, networking researchers, and the IETF are not making progress toward an application-friendly Internet . . .

. . . so software researchers have to do it.

WHAT SHOULD BE MODELED?

HOW SHOULD IT BE MODELED?

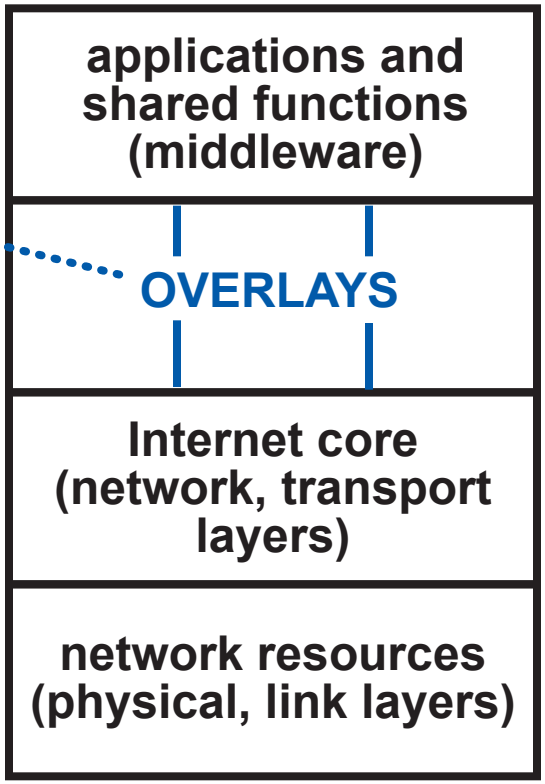
THE IMPORTANCE OF OVERLAYS

Common definition: An *overlay* is a custom-built network layer deployed over existing layers.

OVERLAYS ARE THE MODULES OF NETWORK ARCHITECTURE

FOR TODAY:

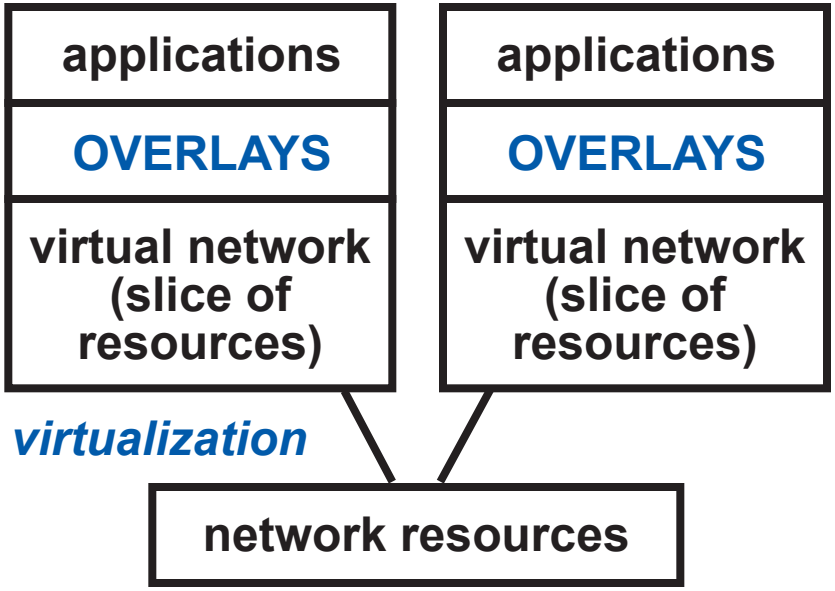
frequently used to support applications better than the unvarnished Internet does



AN OVERLAY IS A "CLEAN SLATE" FOR DESIGN

FOR THE FUTURE:

can experiment with new architectural ideas



in the end, there may be no universal Internet layer *[Roscoe 06]*

A BETTER DEFINITION OF OVERLAYS

[Day 08]

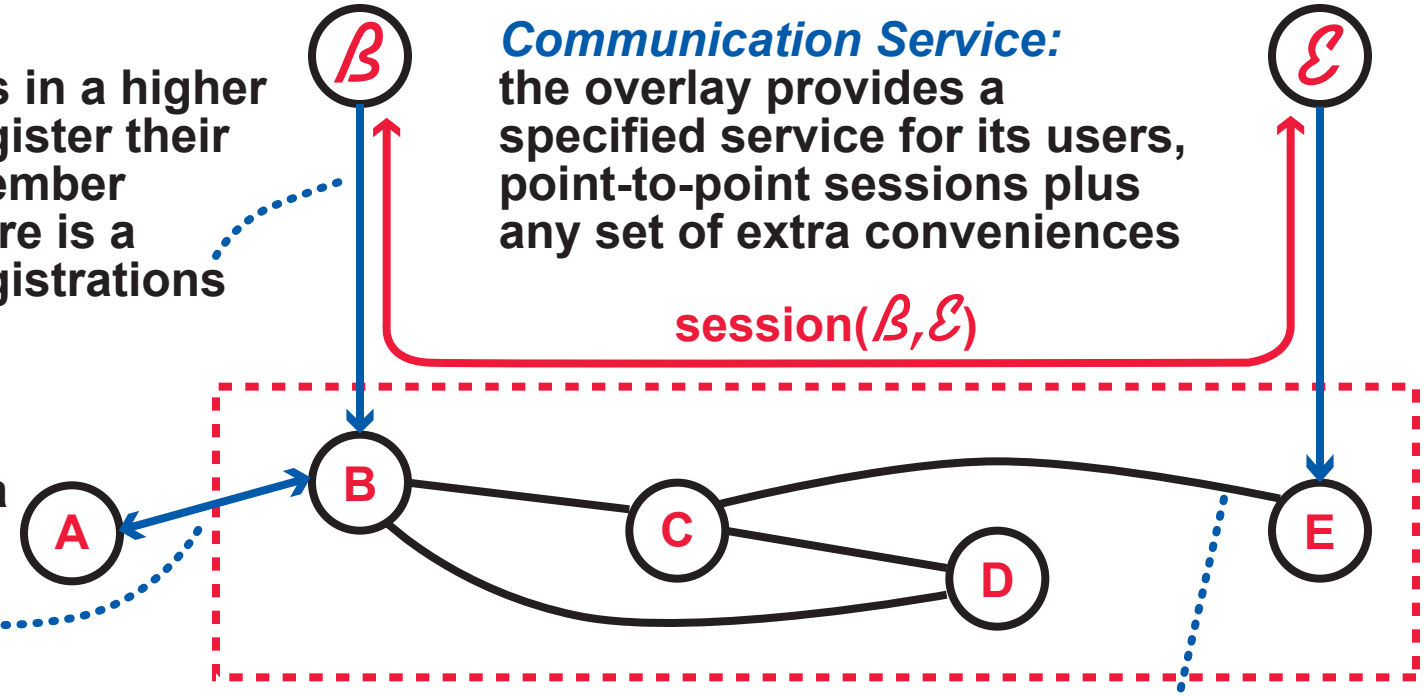
An overlay contains (potentially) all network functions.

There are many instances of the overlay type, varying in their rank (level) and scope (membership).

Overlays are arranged in a "uses" hierarchy.

Registration:
user processes in a higher overlay can register their locations at member processes; there is a directory of registrations

Communication Service:
the overlay provides a specified service for its users, point-to-point sessions plus any set of extra conveniences



Membership:
the members are processes; each has a unique and persistent name from the name space; enrollment protocol accepts and names new members

Routing:
any member can reach any other through a path in the overlay; routing protocol spreads knowledge of links and paths; forwarding protocol uses path knowledge

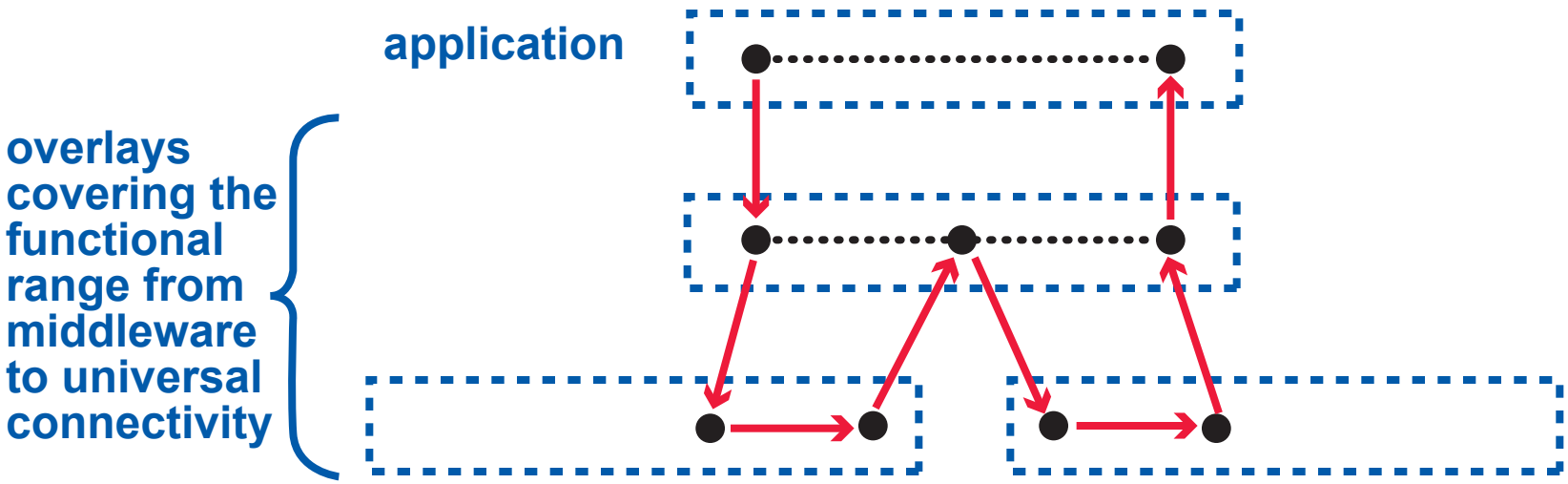
Links:
there is a link between two member processes if both are registered in the same lower overlay

SECURITY AND RESOURCE MANAGEMENT THROUGHOUT

QUESTIONS WE CAN NOW ASK

1 What communication services can be implemented in composable overlays?

2 How can these services be specified and verified?



3

Is there a uniform approach to scaling, by splitting and bridging domains, without the bad side-effects of NAT?

4

Is there a uniform approach to network and application security that allows policies appropriate to each application, unlike firewalls?

5

Are there principles for organizing uses hierarchies?

When and how are overlays bound together?

MODELING THE INTERNET: OUTLINE

WHY SHOULD IT BE MODELED?

Because businesses, networking researchers, and the IETF are not making progress toward an application-friendly Internet . . .

. . . so software researchers have to do it.

WHAT SHOULD BE MODELED?

aspects of overlays, to gain insight

HOW SHOULD IT BE MODELED?

LIGHTWEIGHT MODELING

DEFINITION

- constructing a very abstract model of the core concepts of a system
- using an analysis tool based on exhaustive enumeration to explore its properties

WHY IS IT "LIGHTWEIGHT"?

- because the model is very abstract in comparison to a real implementation, and focuses only on core concepts, it is small and can be constructed quickly
- because the analysis tool is "push-button", it yields results with little effort

in contrast,
theorem proving is not "push-button"

WHAT IS ITS VALUE?

- it is a design tool that reveals conceptual errors early

decades of research on software engineering proves that the cost of fixing a bug rises exponentially with the delay in its discovery

- it is a documentation tool that provides complete, consistent, and unambiguous information to implementors and users
- it is easy (at least to get started) and surprising (you get the result of scenarios you would *never* expect)

"If you like surprises, you will love lightweight modeling."
—Pamela Zave

- EASY + SURPRISING = FUN

WHAT IS THE HIDDEN CHALLENGE?

It is so easy to write a model, ask the analyzer a question, get an answer . . .

. . . but not so easy to know what any of these means in the real world.

NONDETERMINISM IN MODEL

- environment choice
- implementation freedom
- system failure
- concurrency

STATEMENTS IN MODEL

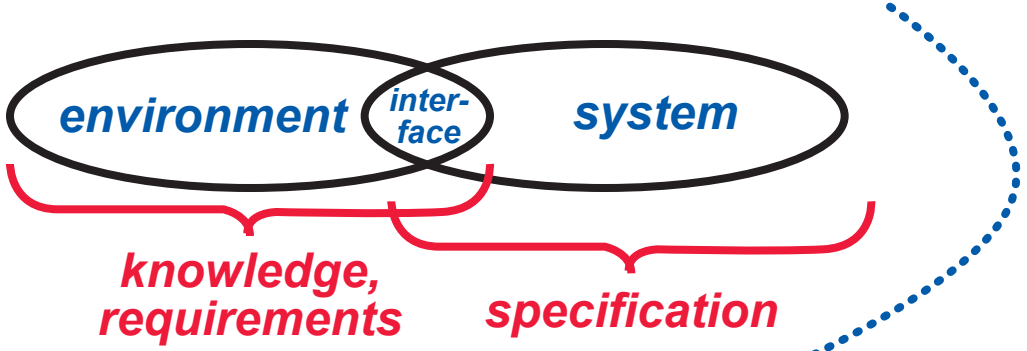
- domain knowledge: description of the environment in which the system will operate (fact or assumption)
- specification: an implementable description of how the hardware/software system should behave
- requirement: a description of how the environment should behave when the system is implemented and deployed
- sanity check: intended to be redundant

ANALYSIS QUESTIONS

- Is the model consistent (can be realized) ?
- Are the knowledge and requirements correct ("validation") ?

sanity checks help

- Is the specification correct ("verification") ?



knowledge & specification => requirements

[Jackson & Zave 95]

IP-BASED VOICE AND MULTIMEDIA APPLICATIONS

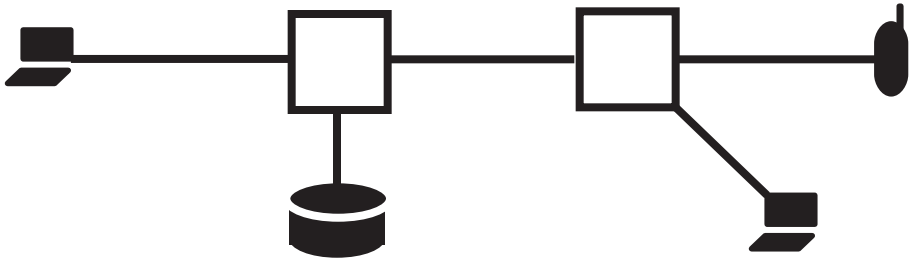
including computer-supported cooperative work, distance learning, emergency services, multiplayer games, and collaborative television

INHERENT PROTOCOL PROBLEMS

PROBLEMS OF SIP (THE SESSION INITIATION PROTOCOL), WHICH IS THE DOMINANT PROTOCOL FOR THESE APPLICATIONS

No clear distinction between endpoints and servers—any server can act as an endpoint—so protocols should be *piecewise*.

SIP is defined *end-to-end*, so servers create unanticipated states.



Endpoints are *peers*—a state-changing event can come from any endpoint, at any time.

SIP is an asymmetric *client-server* protocol.

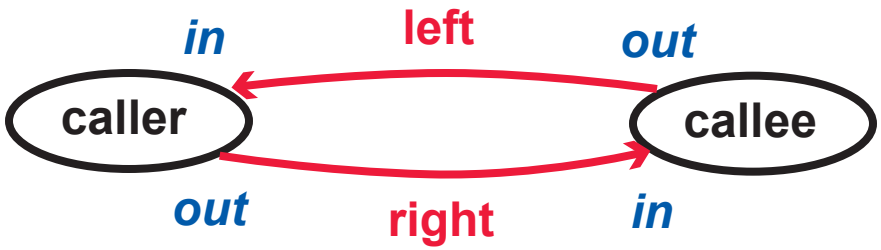
There is a high degree of *concurrency*.

SIP can run over TCP (reliable, FIFO, duplicate-free transport) or over UDP (none of the above), so the protocol must handle message loss and re-ordering.

MODELING IN PROMELA:

THE MODELING LANGUAGE OF THE SPIN MODEL CHECKER

- unlike most mature model checkers, Spin is intended for software, not hardware
- Promela represents processes with variables and arrays, guarded commands, program-like control structures
- inter-process communication uses bounded channels (FIFO, rendezvous, etc. are options)



easy to model protocols, networking, distributed applications

- can place assertions, e.g., "assert(count >= 1)", in-line
- can specify global properties in linear-time temporal logic, including both safety and progress

```
proctype caller (chan in, out) {
            out!invite;
inviting:   do
            :: in?accept; goto confirmed
            :: in?reject; goto end
            od;
confirmed:  do
            :: in?invite; out!accept
            :: out!invite; in?accept
            od;
end:        skip }
```

finite-state-machine style

read executable iff. channel *in* is not empty and its first message is of type *reject*

write executable iff. channel *out* is not full and accepts messages of type *invite*

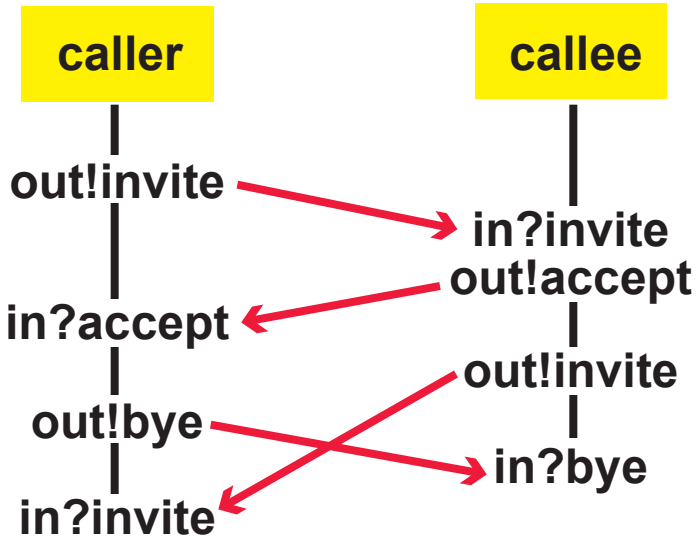
THE SPIN MODEL CHECKER

[Holzmann 2004]

AN EXCELLENT TOOL

- freely available and actively maintained
- well-engineered and mature, with many analysis options
- large user base, in academia, industry, and government (U.S. space program)

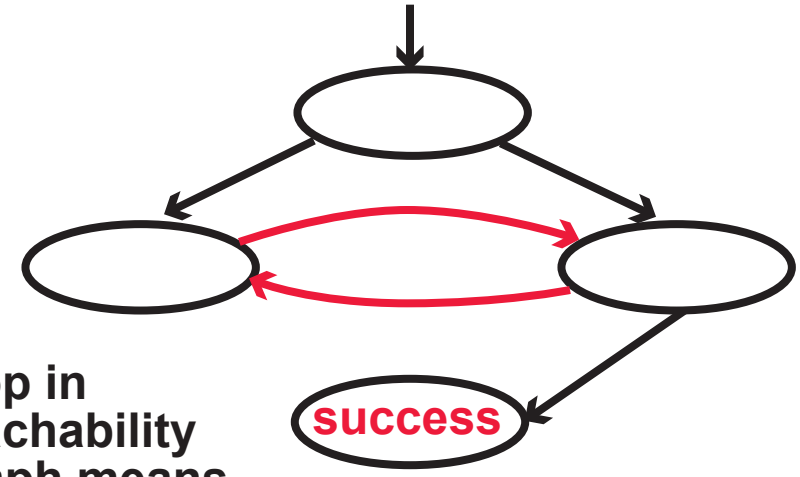
WHEN SPIN DETECTS AN ERROR, THE TRACE (COUNTEREXAMPLE) IS RECORDED, AND SPIN DRAWS A (REALISTIC) MESSAGE-SEQUENCE CHART OF IT



WHAT SPIN CHECKS, FOR ALL POSSIBLE TRACES:

- deadlocks
- invalid end states
- in-line assertions
- temporal-logic assertions

mathematically, a progress property can only be falsified by an infinite trace . . .
. . . so how does Spin do it?



loop in reachability graph means there is an infinite trace that does not reach success, therefore . . .

! (◇ success)

STUDYING SIP WITH SPIN

MODELING ENDPOINTS *[Zave 2008]*

- models show what an endpoint must do to use and interpret the protocol correctly—this is far more complicated than previously understood
- on TCP vs. UDP: with non-FIFO communication, the reachability graph is 100 times the size of the FIFO reachability graph
- an RFC documents 7 race conditions—our model reveals those and 42 others of the same type

ABSTRACTIONS *[Zave et al. 2009]*

- our "StratoSIP" language provides generality while using only an efficient subset of SIP
- compilation of StratoSIP into SIP signaling verified correct with Spin

MODELING SERVERS *[Bond et al. 2010]*

- provides the first rigorous definition of "transparent" behavior, which is the foundation of all other behaviors
- we modified Spin to generate test cases automatically, yielding comprehensive test sets

DOCUMENTING SIP

- we annotate our models with pointers to the relevant sections of RFCs
- as documentation, our models are guaranteed to be complete, consistent, and unambiguous
- also, you know where to find the answer to your question!
- the final frontier—influencing the IETF

DISTRIBUTED HASH TABLES (DHTs)

ROLE AND POPULARITY

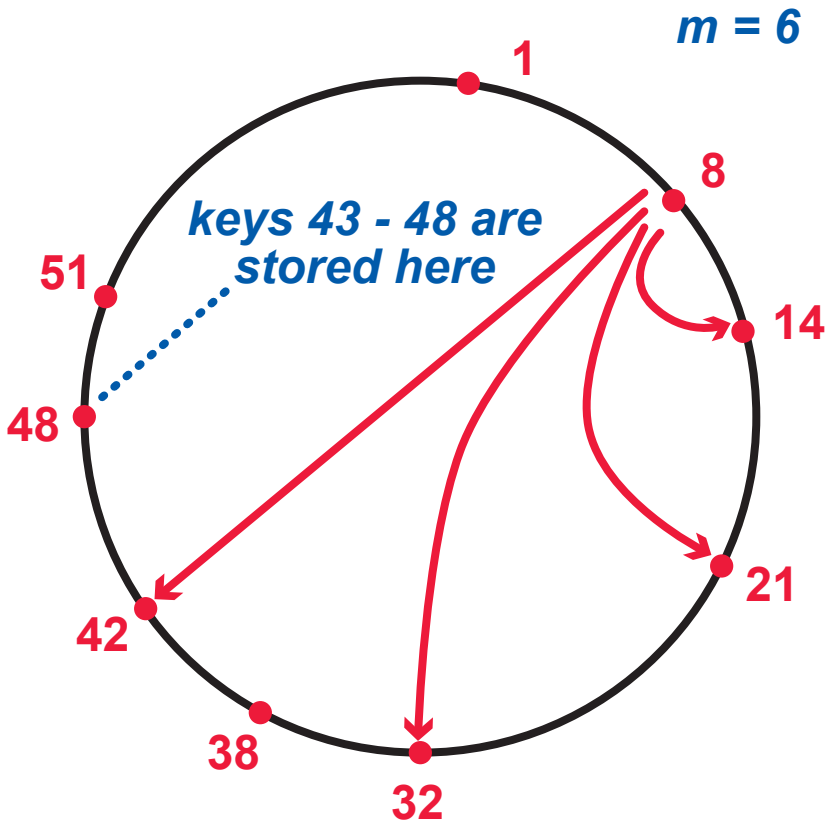
- perform storage and lookup of (key,value) pairs
- data is distributed across thousands or millions of participating nodes
- widely used in peer-to-peer applications, for data structures such as directories

CHORD IS THE BEST-KNOWN, MOST STUDIED DHT

"Three features that distinguish Chord from many peer-to-peer lookup protocols are its simplicity, *provable correctness,* and provable performance."

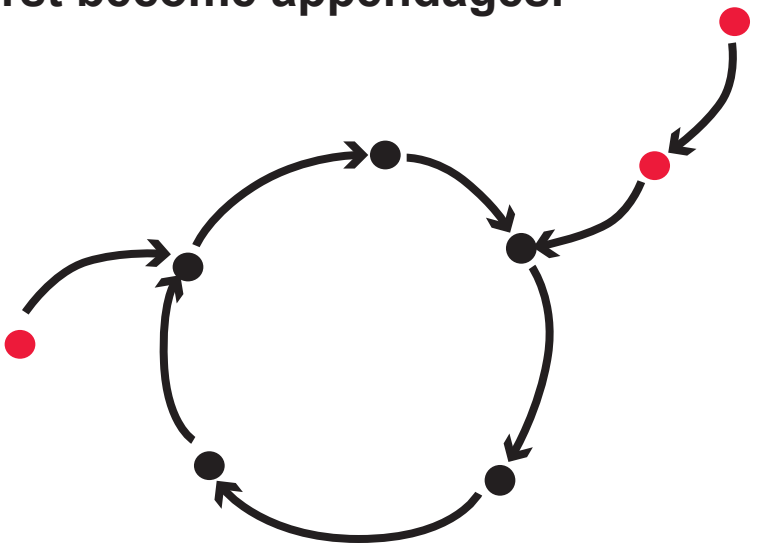
TYPICAL STRUCTURE

- identifier of a node (assumed unique) is an m-bit hash of its IP address
- members are arranged in a ring, with each member node having a successor pointer to the next member node

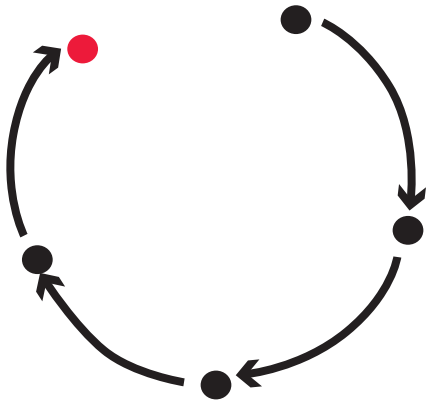


THE RING-MAINTENANCE PROTOCOL

When nodes *join*, they first become appendages.

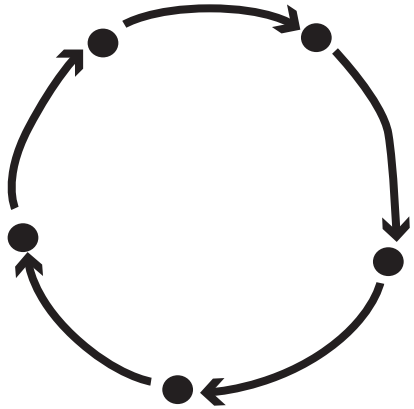


When nodes *fail* (or leave), they make gaps in the ring.



protocol has redundant pointers and operations that run regularly at each node

its purpose is to maintain all the pointers and repair disruptions of the ring



Understanding this protocol means understanding what properties are preserved by joins and failures.

These invariants are the basis for making repairs.

Finding the invariants is very difficult, especially after-the-fact.

as you will see, the designers of Chord did not understand them

It requires a lot of trial and error.

MODELING IN ALLOY

[Jackson 06]

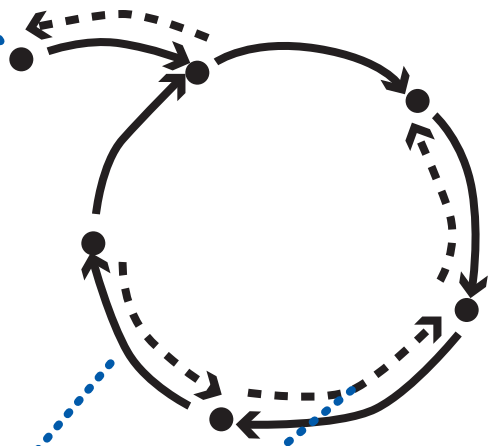
The language is a streamlined blend of predicate logic and relational algebra.

Although Alloy is first-order, second-order quantification can be simulated.

STATE OF A CHORD NETWORK

Node is a basic type

total ordering on nodes represents identifier order

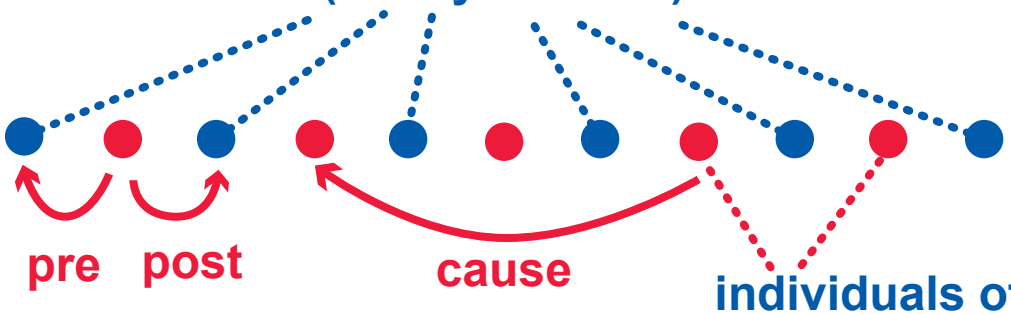


succ and prdc are relations from Node to Node, with uniqueness constraints

REPRESENTING TIME

All state information is time-stamped.

individuals of type Time (totally ordered)



individuals of type Event

STATE INVARIANTS

a cycle node can reach itself by following succ pointers

```
pred OneOrderedCycle[t: Time] {
    let cycleMembers =
        { n: Node | n in n.(^(succ.t)) } |

    some cycleMembers

&& (all disj n1, n2: cycleMembers |
        n1 in n2.(^(succ.t)) )

&& (all disj n1, n2, n3: cycleMembers |
        n2 = n1.succ.t => ! Between[n1,n3,n2] ) }
```

at least one cycle

at most one

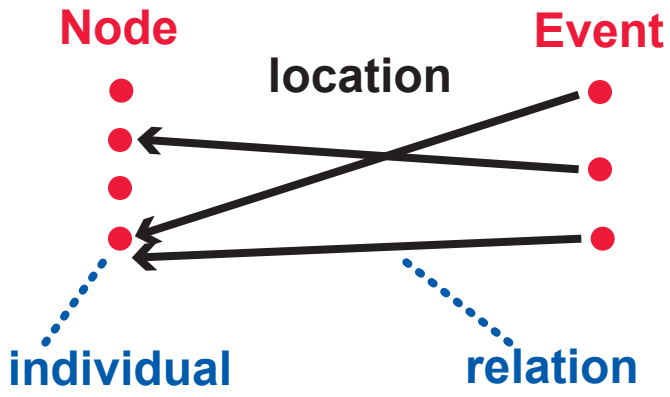
cycle is globally ordered

THE ALLOY ANALYZER

The Alloy Analyzer uses "model enumeration" to check all instances of a model within a finite scope.

finite scope: there is a limit on the members of each basic type

As with model checking, when a check fails, there is a counterexample to examine.



An instance might encode a trace, but time and events are not treated differently from individuals of any other types.

PROOF OF A PROGRESS PROPERTY

Theorem: In any reachable state, if there are no subsequent disruptions, then eventually the network will become ideal and remain ideal.

1 Use Analyzer to establish an invariant—a property preserved by all events; this characterizes the reachable states.

analyze traces of one or two events to show that property is true after if true before

2 Use Analyzer to show that any time the state is not ideal, some improving event is enabled.

3 Show that a finite number of improvements will make a finite network ideal.

this non-Alloy step guarantees progress

4 Use Analyzer to show that any time the state is ideal, no "improving" event is enabled.

analyzable scope far exceeds the largest scope where any new phenomena appears

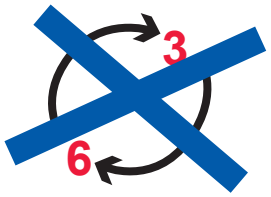
CHORD PROPERTIES CLAIMED INVARIANT

[Zave 10a]

NOT ONE of these properties is actually an invariant!

and the protocol is NOT correct!

AtMostOneCycle



AtLeastOneCycle

OrderedAppendages

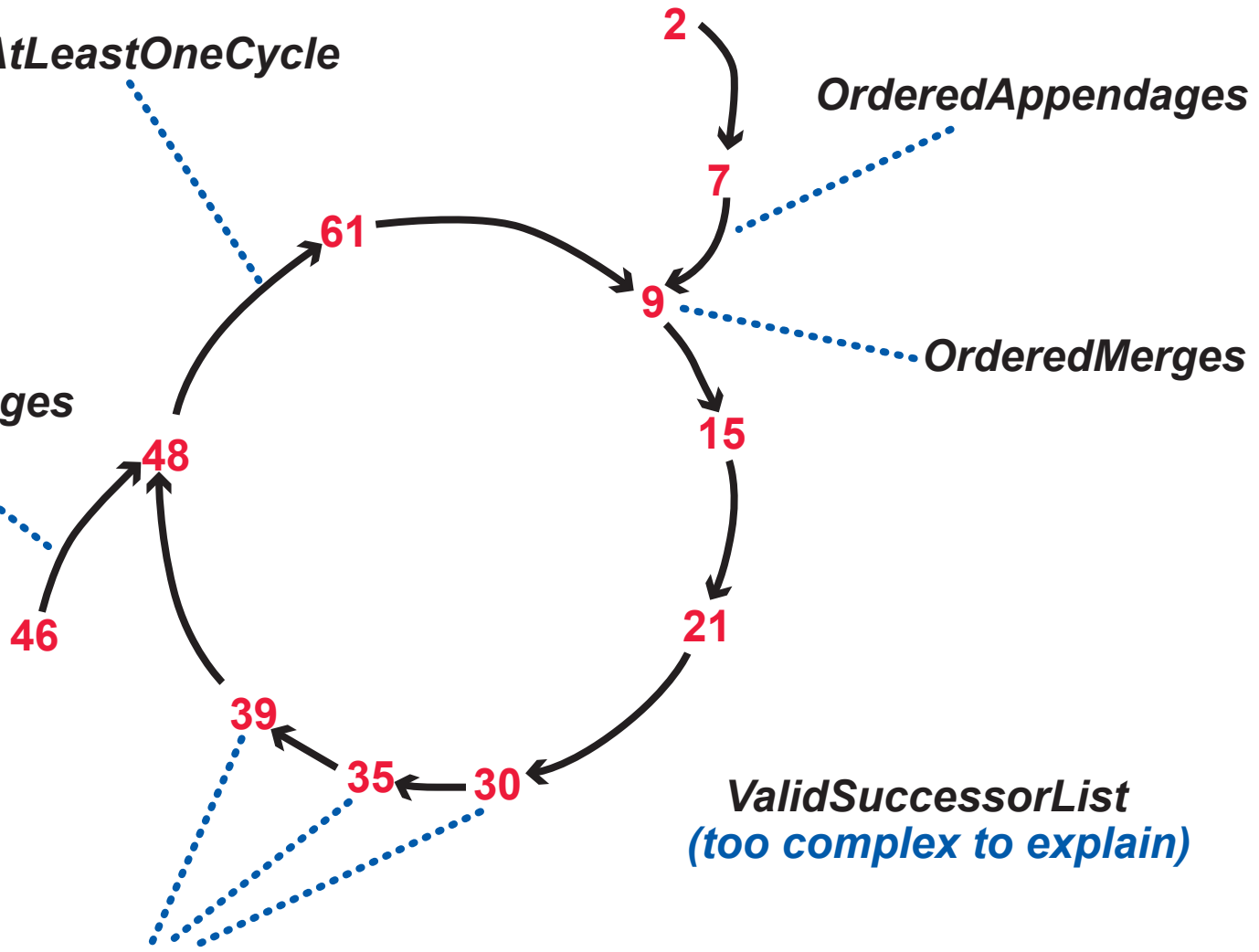
OrderedMerges

ConnectedAppendages

ValidSuccessorList
(too complex to explain)

OrderedCycle

two studies of Chord have been made using model checking, and they did not find any of these problems



USE THE RIGHT LANGUAGE FOR YOUR PURPOSE

	PROMELA/SPIN	ALLOY
model state	small and bounded	small and bounded
reachable state space	automatically generated, exact, not readable	invariant is a user-constructed superset; readable
temporal sequencing	built into Promela; displayed well by Spin	not built into Alloy language
temporal logic	Spin automatically checks any formula in temporal logic	Alloy Analyzer can only check safety properties on short traces
state structure	primitive in Promela; displayed poorly by Spin	Alloy language is rich and expressive; many display options
state assertions	all but the most basic ones must be written as C programs, which is awkward and difficult	Alloy language is rich, expressive, and concise

the price of push-button analysis

MODELING THE INTERNET: SUMMARY

WHY SHOULD IT BE MODELED?

Because businesses, networking researchers, and the IETF are not making progress toward an application-friendly Internet . . .

. . . so software researchers have to do it.

WHAT SHOULD BE MODELED?

aspects of overlays, to gain insight

HOW SHOULD IT BE MODELED?

using lightweight modeling tools

with attention to the real-world significance of properties, not just their formal semantics

using the right language for each purpose

NO SOFTWARE SYSTEM IS MORE IMPORTANT THAN THE INTERNET

THIS IS A TIME OF GREAT RESEARCH OPPORTUNITY

FURTHER READING

[Bond et al. 10] Gregory W. Bond *et al.*, Specification and evaluation of transparent behavior for SIP back-to-back user agents, *Proc. 4th IPTComm*, 2010.

[Clark et al. 05] David D. Clark *et al.,* Tussle in cyberspace: Defining tomorrow's Internet, *IEEE/ACM Transactions on Networking,* June 2005.

[Handley 06] Mark Handley, Why the Internet only just works, *BT Technology Journal,* July 2006.

[Day 08] John Day, *Patterns in Network Architecture*, Prentice Hall, 2008.

[Holzmann 04] Gerard J. Holzmann, *The Spin Model Checker: Primer and Reference Manual*, Addison-Wesley, 2004.

[Jackson 06] Daniel Jackson, *Software Abstractions: Logic, Language, and Analysis,* MIT Press, 2006.

[Jackson & Zave 95] Michael Jackson and Pamela Zave, Deriving specifications from requirements: An example, *Proc. 17th Intl. Conf. on Software Engineering,* 1995.

[Rexford 10] Jennifer Rexford, Notes for a course on Advanced Computer Networks, Princeton University, 2010.

[Roscoe 06] Timothy Roscoe, The end of Internet architecture, *Proc. 5th Workshop on Hot Topics in Networks,* 2006.

FURTHER READING, CONTINUED

[Zave 08] Pamela Zave, Understanding SIP through model-checking, *Proc. 2nd IPTComm,* 2008.

[Zave et al. 09] Pamela Zave *et al.,* Abstractions for programming SIP back-to-back user agents, *Proc. 3rd IPTComm,* 2009.

[Zave 10a] Pamela Zave, Lightweight modeling of network protocols in Alloy, 2010.

[Zave 10b] Pamela Zave, Internet evolution and the role of software engineering, *Proc. Symp. on the Future of Software Engineering, Springer LNCS,* 2010.

www2.research.att.com/~pamela