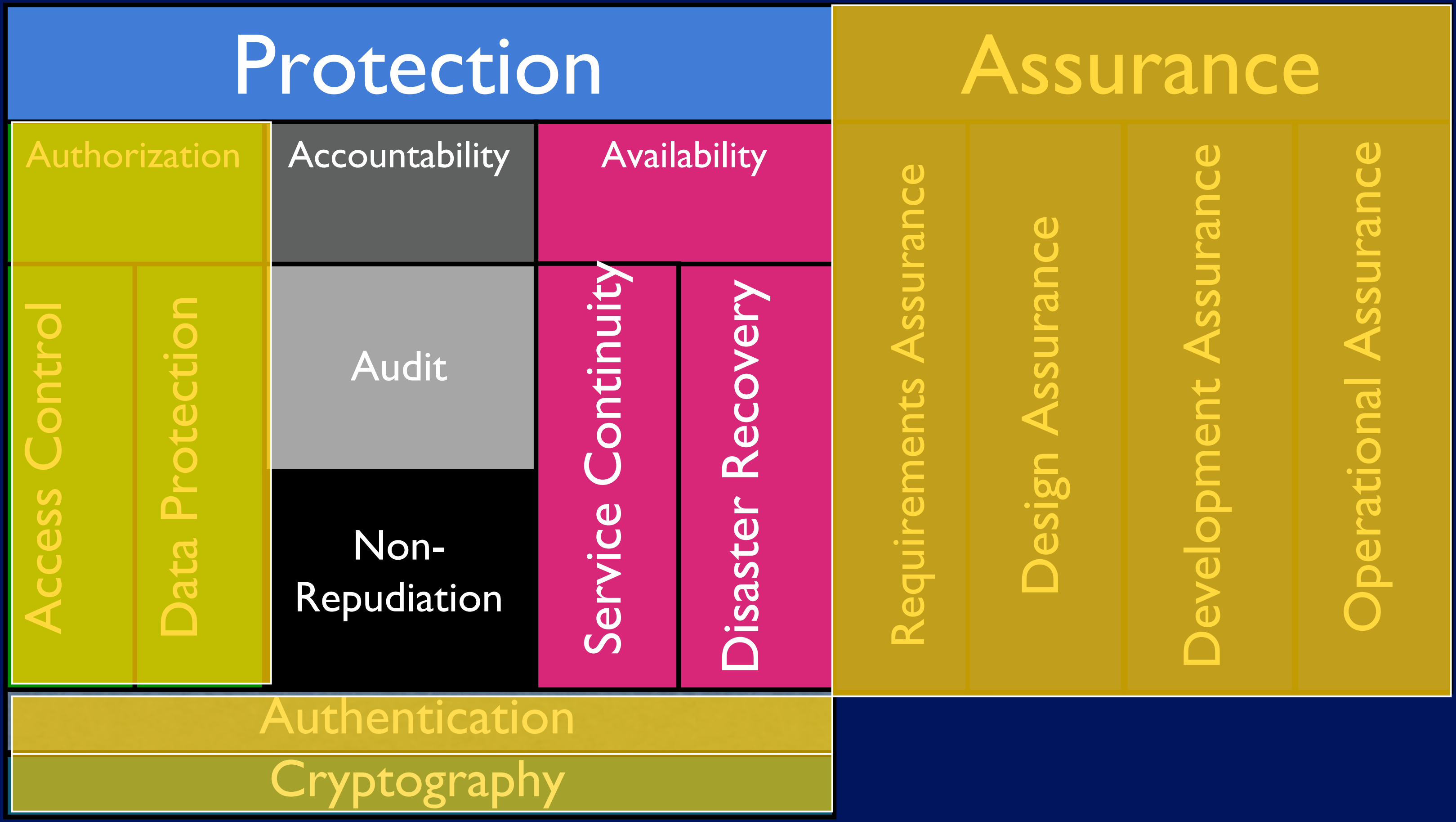
Accountability and Availability

Secure Application Development

Module 6

Konstantin Beznosov

Where We Are



Protection

Authorization

Access Control

Data Protection

Accountability

Audit

Non-Repudiation

Availability

Service Continuity

Disaster Recovery

Authentication

Cryptography

Assurance

Requirements Assurance

Design Assurance

Development Assurance

Operational Assurance

Outline

- Accountability
  - What is auditing?
  - What does an audit system look like?
  - How do you design an auditing system?
  - Auditing mechanisms
- Availability
  - in the presence of failures
    - FT terminology
    - k fault tolerance
    - two army problem
    - Byzantine Generals problem
    - Services continuity and disaster recovery
  - in the presence of attacks
    - Failures vs. attacks
    - Random vs. scale-free networks
    - Internet tolerance to attacks and failures
    - Services continuity and disaster recovery

Accountability

What is Auditing?

- Logging
  - Recording events or statistics to provide information about system use and performance
- Auditing
  - Analysis of log records to present information about the system in a clear, understandable manner

What’s Auditing Good For?

- Describing security state
  - Determine if system enters unauthorized state
- Evaluating effectiveness of protection mechanisms
  - Determine which mechanisms are appropriate and working
  - Deter attacks because of presence of record

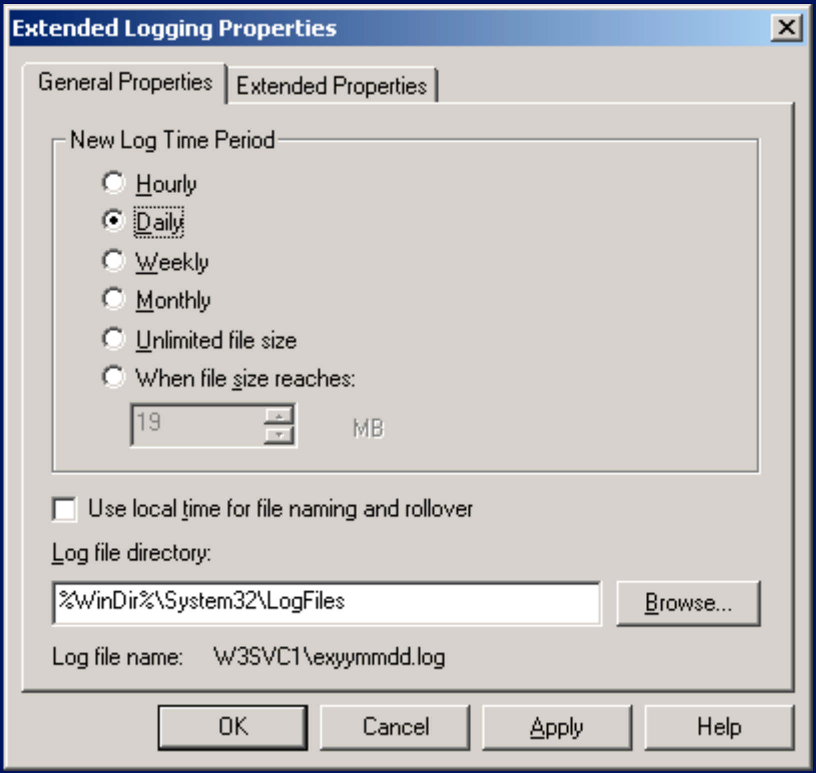
Problems

- What do you log?
  - Hint: looking for violations of a policy, so record at least what will show such violations
- What do you audit?
  - Need not audit everything
  - Key: what is the policy involved?

Audit System Structure

- Logger
  - Records information, usually controlled by parameters
- Analyzer
  - Analyzes logged information looking for something
- Notifier
  - Reports results of analysis

Example: Logging Configuration in IIS



New Log Time Period

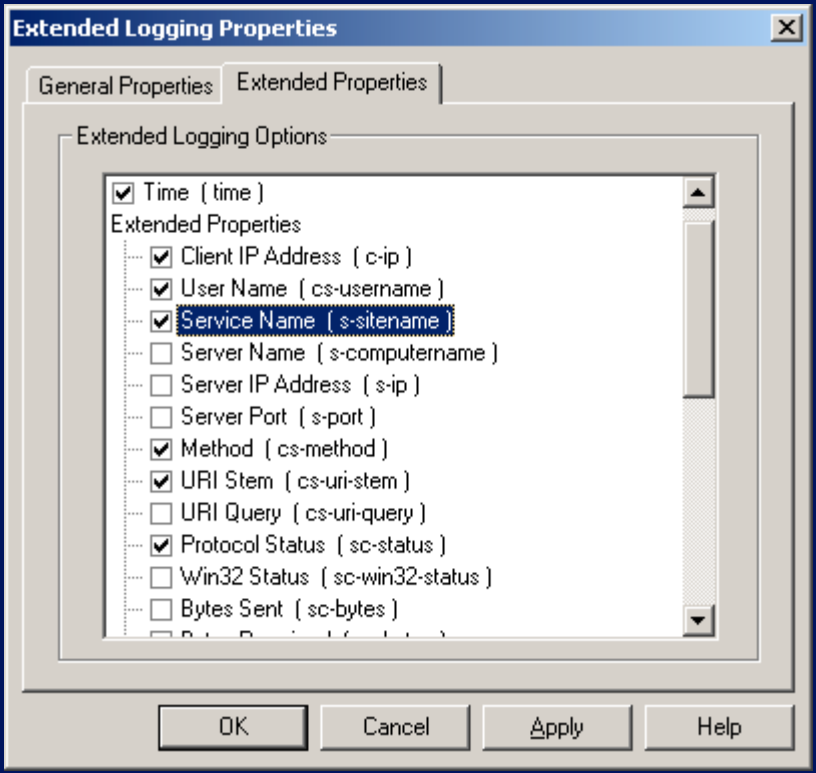
○ Hourly
● Daily
○ Weekly
○ Monthly
○ Unlimited file size
○ When file size reaches:

19 MB

□ Use local time for file naming and rollover

Log file directory: %WinDir%\System32\LogFiles

Log file name: W3SVC1\exyymmdd.log



Extended Logging Options

☑ Time (time)

Extended Properties

- ☑ Client IP Address (c-ip)
- ☑ User Name (cs-username)
- ☑ Service Name (s-sitename)
- □ Server Name (s-computername)
- □ Server IP Address (s-ip)
- □ Server Port (s-port)
- ☑ Method (cs-method)
- ☑ URI Stem (cs-uri-stem)
- □ URI Query (cs-uri-query)
- ☑ Protocol Status (sc-status)
- □ Win32 Status (sc-win32-status)
- □ Bytes Sent (sc-bytes)

Example: RACF

- Security enhancement package for IBM’s MVS/VM
- Logs failed access attempts, use of privilege to change security levels, and (if desired) RACF interactions
- View events with LISTUSERS commands

RACF: Sample Entry

```
USER=EW125004   NAME=S.J.TURNER   OWNER=SECADM   CREATED=88.004
  DEFAULT-GROUP=HUMRES    PASSDATE=88.004   PASS-INTERVAL=30
  ATTRIBUTES=ADSP
  REVOKE DATE=NONE    RESUME-DATE=NONE
  LAST-ACCESS=88.020/14:15:10
  CLASS AUTHORIZATIONS=NONE
  NO-INSTALLATION-DATA
  NO-MODEL-NAME
  LOGON ALLOWED     (DAYS)  (TIME)
  --------------------------------
  ANYDAY                    ANYTIME
    GROUP=HUMRES AUTH=JOIN CONNECT-OWNER=SECADM
                                   CONNECT-DATE=88.004
      CONNECTS= 15  UACC=READ LAST-CONNECT=88.018/16:45:06
      CONNECT ATTRIBUTES=NONE
      REVOKE DATE=NONE RESUME DATE=NONE
    GROUP=PERSNL AUTH=JOIN CONNECT-OWNER=SECADM CONNECT-DATE:88.004
      CONNECTS= 25 UACC=READ LAST-CONNECT=88.020/14:15:10
      CONNECT ATTRIBUTES=NONE
      REVOKE DATE=NONE RESUME DATE=NONE
    SECURITY-LEVEL=NONE SPECIFIED
    CATEGORY AUTHORIZATION
       NONE SPECIFIED
```

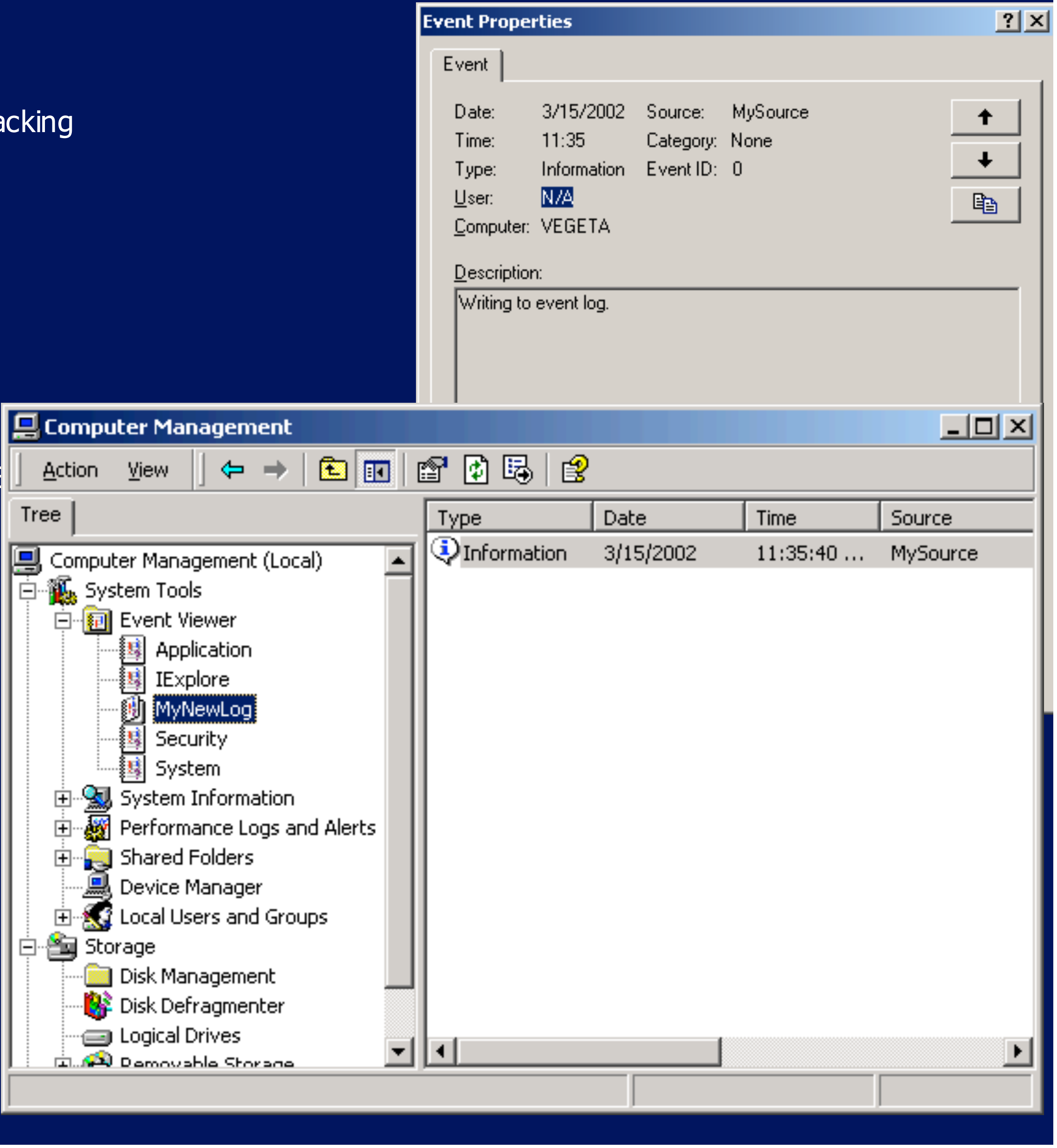
Example: Windows NT

- Different logs for different types of events
  - *System event* logs record system crashes, component failures, and other system events
  - *Application event* logs record events that applications request be recorded
  - *Security event* log records security-critical events such as logging in and out, system file accesses, and other events
- Logs are binary
  - use *event viewer* to see them
- If log full, can have
  - system shut down,
  - logging disabled, or
  - logs overwritten

Windows NT Sample Entry

```
Date:         2/12/2000   Source:     Security
Time:         13:03       Category:   Detailed Tracking
Type:         Success     EventID:    592
User:         WINDSOR\Administrator
Computer:     WINDSOR

Description:
A new process has been created:
    New Process ID:       2216594592
    Image File Name:
    \Program Files\Internet Explorer\IEXPLORE.EXE
    Creator Process ID:   2217918496
    User Name:            Administrator
    FDomain:              WINDSOR
    Logon ID:             (0x0,0x14B4c4)
```



Event Properties

Date:	3/15/2002	Source:	MySource
Time:	11:35	Category:	None
Type:	Information	Event ID:	0
User:	N/A		
Computer:	VEGETA		

Description:

Writing to event log.

Computer Management

Type	Date	Time	Source
Information	3/15/2002	11:35:40 ...	MySource

Analyzer

- Analyzes one or more logs
  - Logs may come from multiple systems, or a single system
  - May lead to changes in logging
  - May lead to a report of an event

Examples

1. Using *swatch* to find instances of *telnet* from *tcpd* logs:

   ```
   /telnet/&!/localhost/&!/*.site.com/
   ```

2. Intrusion detection analysis engine (director)
   - Takes data from sensors and determines if an intrusion is occurring

Notifier

- Informs analyst, other entities of results of analysis
- May reconfigure logging and/or analysis on basis of results

Examples

1. Using *swatch* to notify of *telnets*

```
/telnet/&!/localhost/&!/*.site.com/      mail staff
```

2. Three failed logins in a row disable user account
   - Notifier disables account, notifies sysadmin

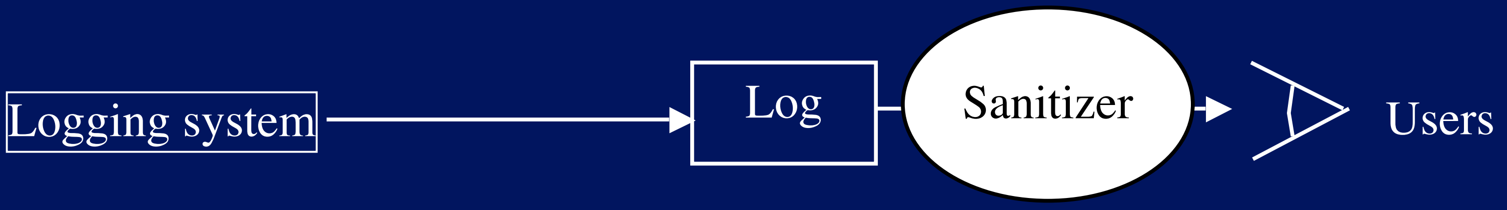
Designing an Audit System

- Essential component of security mechanisms
- Goals determine what is logged
    - Idea: auditors want to detect violations of policy, which provides a set of constraints that the set of possible actions must satisfy
    - So, audit functions that may violate the constraints
- Constraint p_i : *action* $\Rightarrow$ *condition*

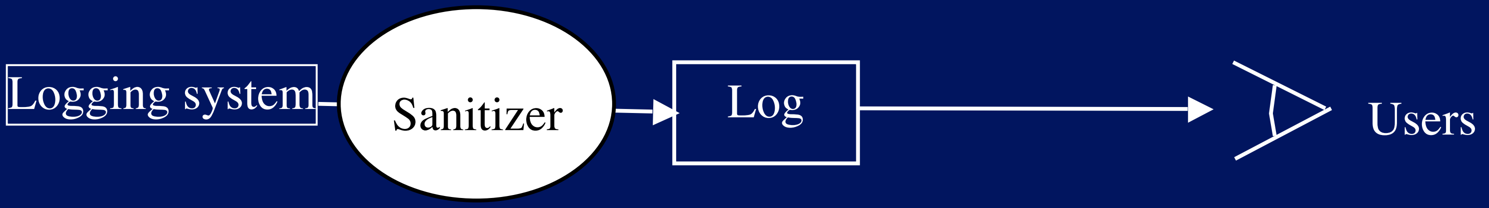
Example: Bell–LaPadula

- Simple security property and *-property
  - S reads $O \Rightarrow L(S) \geq L(O)$
  - S writes $O \Rightarrow L(S) \leq L(O)$
  - To check for violations, on each read and write, must log $L(S)$, $L(O)$, action (read, write), and result (success, failure)
- Note: need *not* record S, O!
  - In practice, done to identify the object of the (attempted) violation and the user attempting the violation
- What about RBAC?

Logging Organization



- prevents information from leaving site
  - Users' privacy not protected from system administrators, other administrative personnel



- prevents information from leaving system
  - Data simply not recorded, or data scrambled before recording

Reconstruction

- *Anonymizing sanitizer* cannot be undone
  - No way to recover data from this
- *Pseudonymizing sanitizer* can be undone
  - Original log can be reconstructed
- Importance
  - Suppose security analysis requires access to information that was sanitized?
- Key: sanitization must preserve properties needed for security analysis

Example

- Company wants to keep its IP addresses secret, but wants a consultant to analyze logs for an address scanning attack
  - Connections to port 25 on IP addresses 10.163.5.10, 10.163.5.11, 10.163.5.12, 10.163.5.13, 10.163.5.14, 10.163.5.15
  - Sanitize with random IP addresses
    - Cannot see sweep through consecutive IP addresses
  - Sanitize with sequential IP addresses
    - Can see sweep through consecutive IP addresses

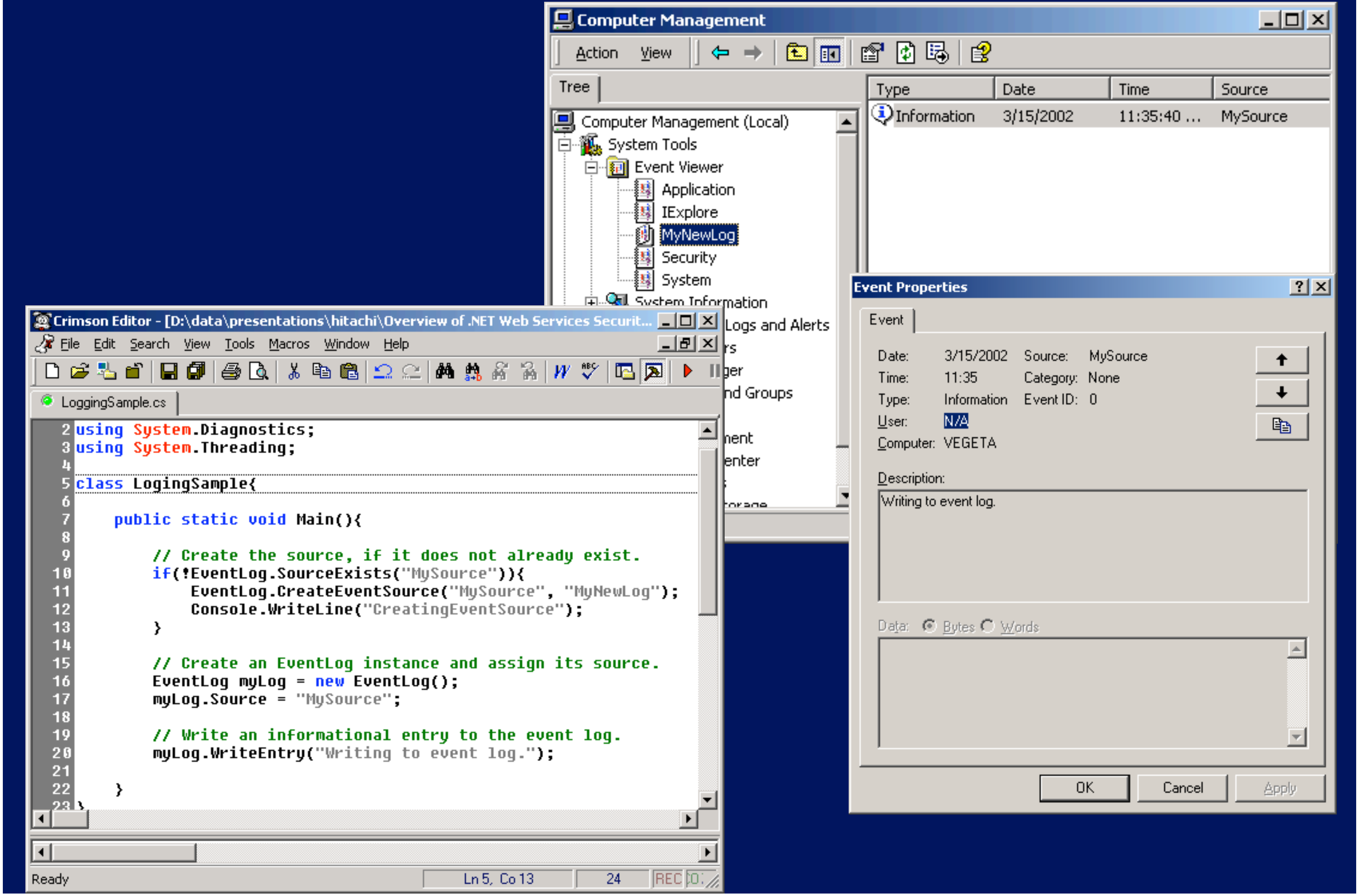
Application Logging

- Applications logs made by applications
  - Applications control what is logged
  - Typically use high-level abstractions such as:

    ```
    su: bishop to root on /dev/ttyp0
    ```

  - Does not include detailed, system call level information such as results, parameters, etc.

Example: Application Logging in .NET



```
using System.Diagnostics;
using System.Threading;

class LogingSample{

    public static void Main(){

        // Create the source, if it does not already exist.
        if(!EventLog.SourceExists("MySource")){
            EventLog.CreateEventSource("MySource", "MyNewLog");
            Console.WriteLine("CreatingEventSource");
        }

        // Create an EventLog instance and assign its source.
        EventLog myLog = new EventLog();
        myLog.Source = "MySource";

        // Write an informational entry to the event log.
        myLog.WriteEntry("Writing to event log.");

    }
}
```

Type	Date	Time	Source
Information	3/15/2002	11:35:40 ...	MySource

Date: 3/15/2002 Source: MySource
Time: 11:35 Category: None
Type: Information Event ID: 0
User: N/A
Computer: VEGETA

Description:
Writing to event log.

Data: ● Bytes ○ Words

System Logging

- Log system events such as kernel actions
  - Typically use low-level events

```
3876 ktrace  CALL    execve(0xbfbff0c0,0xbfbff5cc,0xbfbff5d8)
3876 ktrace  NAMI    "/usr/bin/su"
3876 ktrace  NAMI    "/usr/libexec/ld-elf.so.1"
3876 su      RET     xecve 0
3876 su      CALL     __sysctl(0xbfbff47c,0x2,0x2805c928,0xbfbff478,0,0)
3876 su      RET     __sysctl 0
3876 su      CALL    mmap(0,0x8000,0x3,0x1002,0xffffffff,0,0,0)
3876 su      RET     mmap 671473664/0x2805e000
3876 su      CALL    geteuid
3876 su      RET     geteuid 0
```

  - Does not include high-level abstractions such as loading libraries (as above)

How Are System and Application Logging Differ?

- Differ in focus
  - Application logging focuses on application events, like failure to supply proper password, and the broad operation (what was the reason for the access attempt?)
  - System logging focuses on system events, like memory mapping or file accesses, and the underlying causes (why did access fail?)
- System logs usually much bigger than application logs
- Can do both, try to correlate them

Key Points on Accountability

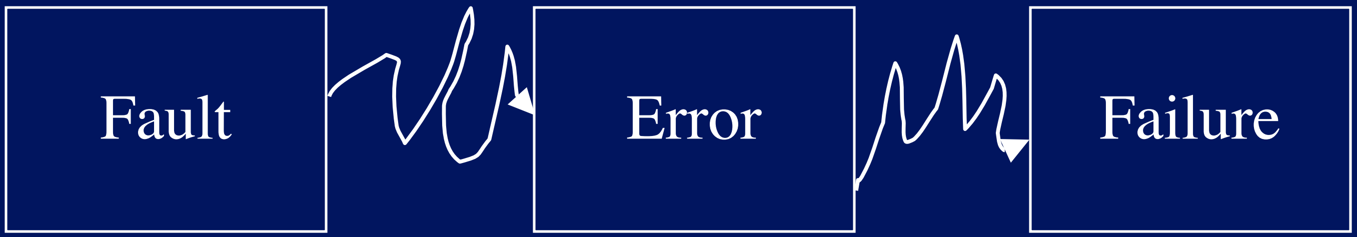
- Logging is collection and recording; audit is analysis
- Need to have clear goals when designing an audit system
- Auditing should be designed into system, not patched into system after it is implemented

Availability

Availability in the Presence of Failures

Failures, Errors, and Faults

- A system is said to fail when it cannot meet its promises
- Error may lead to a failure
- Fault -- a cause of an error



Fault Types

- Transient: occur once and then disappear

- Intermittent: occurs, then vanishes, then reappears

- Permanent: continues to exist

Availability and Reliability

- **Availability**: Probability that a system operates correctly at any given moment and is available to perform its functions

- **Reliability**: time period during which a system continues to be available to perform its functions
  - Mean Time to Failure (MTTF)

- Problem: calculate system availability and reliability if it's unavailable for 1 second every hour.

Fault Tolerance

A fault tolerant system can provide its services even in the presence of faults

Classification of Failure Modes

Type of failure	Description
Crash failure	A server halts, but is working correctly until it halts
Omission failure	A server fails to respond to incoming requests
Receive omission	A server fails to receive incoming messages
Send omission	A server fails to send messages
Timing failure	A server's response lies outside the specified time interval
Response failure	The server's response is incorrect
Value failure	The value of the response is wrong
State transition failure	The server deviates from the correct flow of control
Arbitrary (a.k.a. Byzantine) failure	A server may produce arbitrary responses at arbitrary times

Achieving k fault tolerance

A system is k fault tolerant if it can survive faults in k components

- silent failure vs. Byzantine failure

  k+1 2k+1

Ways to Deal with Failures

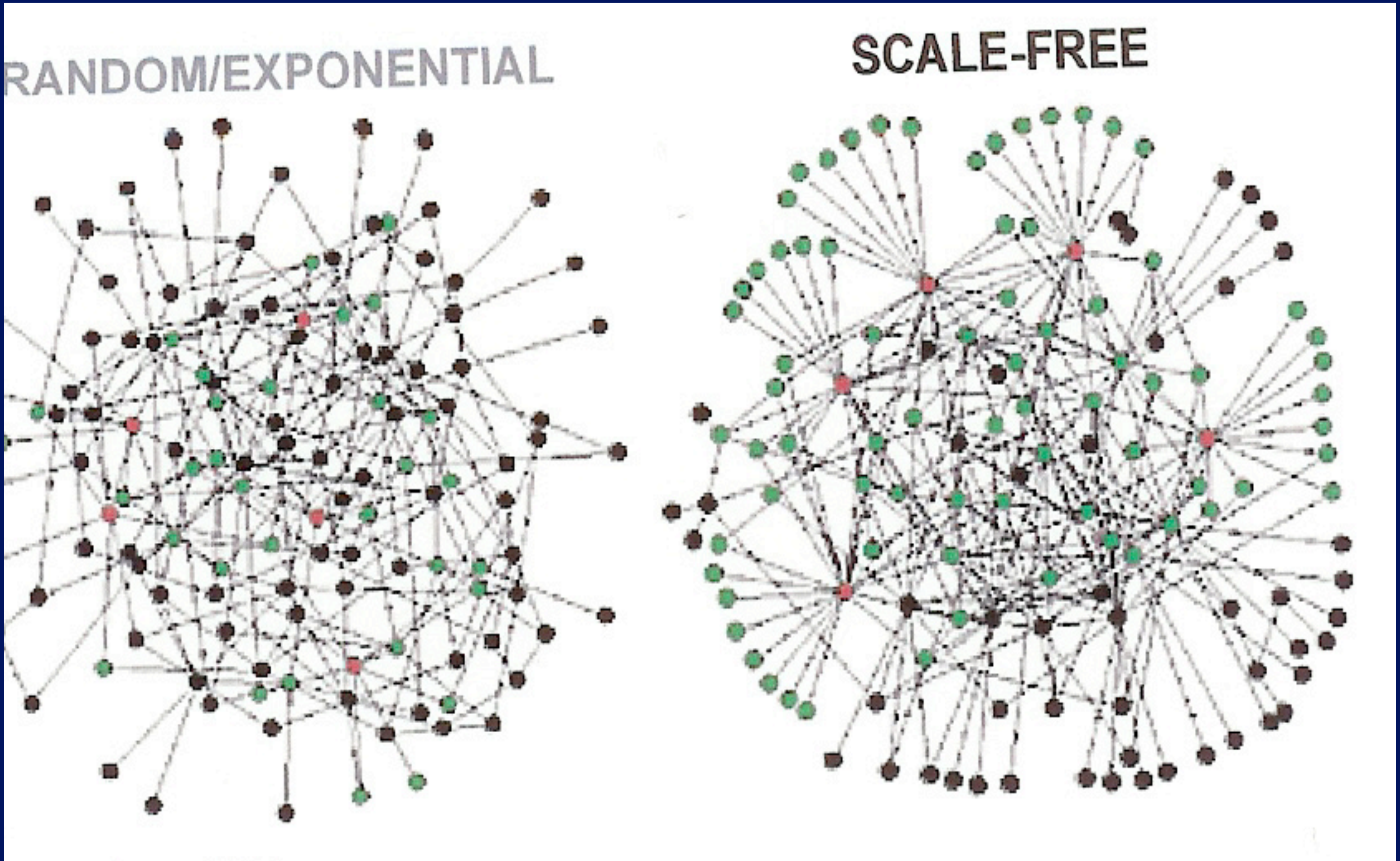
- Service continuity
  - Masking failures via
    - Redundancy of
      - information
      - time
      - physical
- Disaster recovery
  - Backward recovery
    - check pointing
  - Forward recovery
    - bringing system into a correct new state
  - Don’t underestimate backups!

Availability in the Presence of Attacks

Failures vs. Attacks

- **Failure**
  - Random unavailability of participants and/or infrastructure elements
- **Attack**
  - Systematic unavailability of participants and/or infrastructure elements

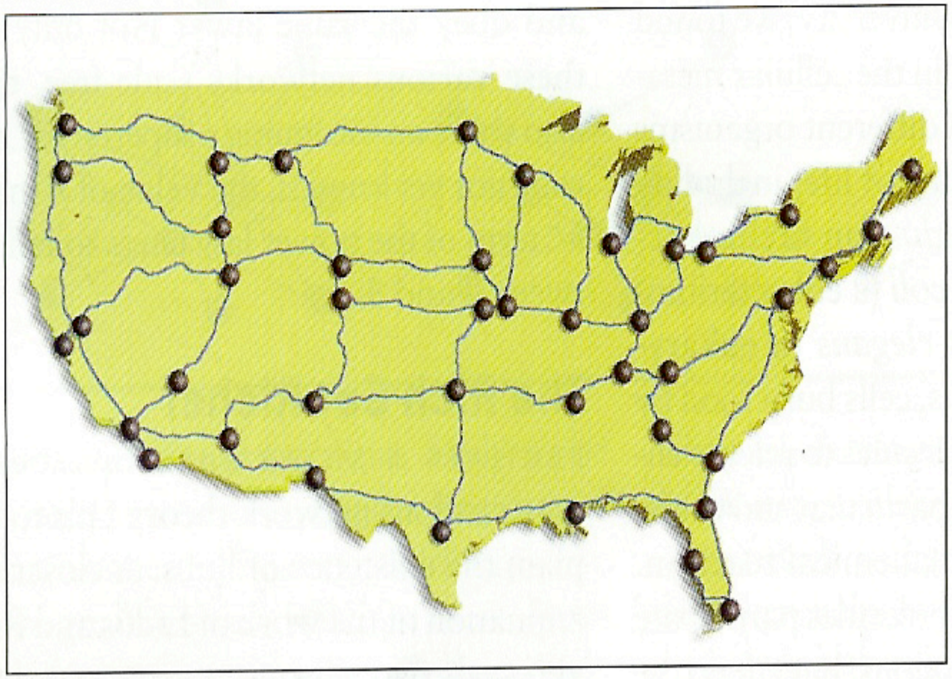
Random vs. Scale-free Networks



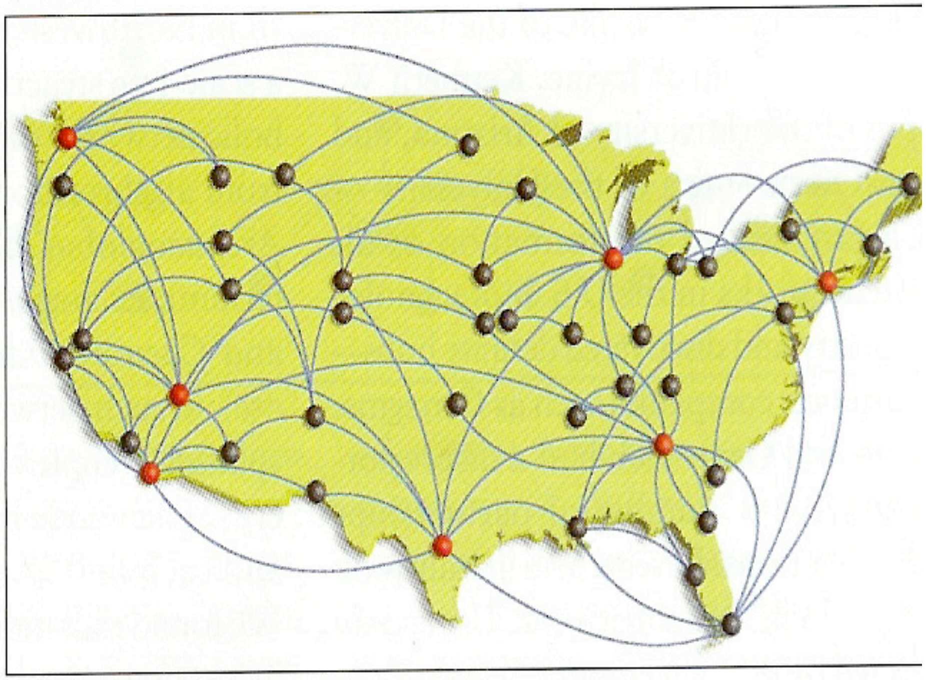
e: the journal Nature

airline system (simplified in right map),

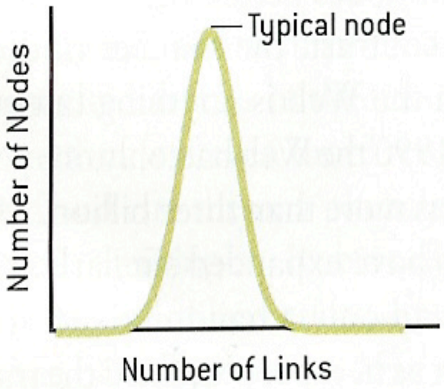
Random Network



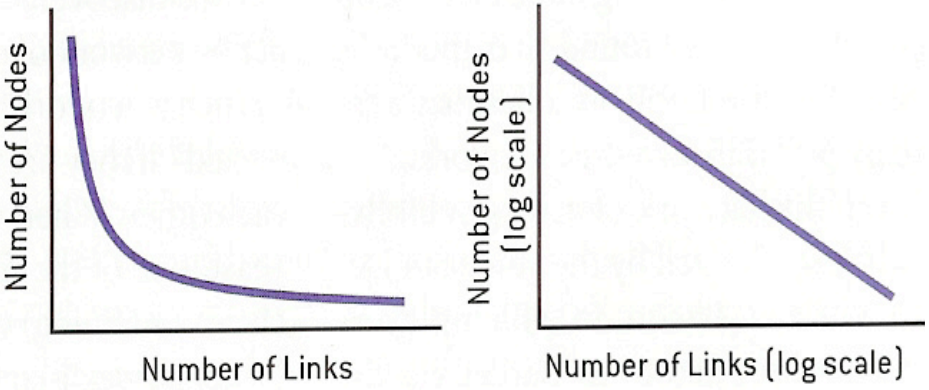
Scale-Free Network



Bell Curve Distribution of Node Linkages

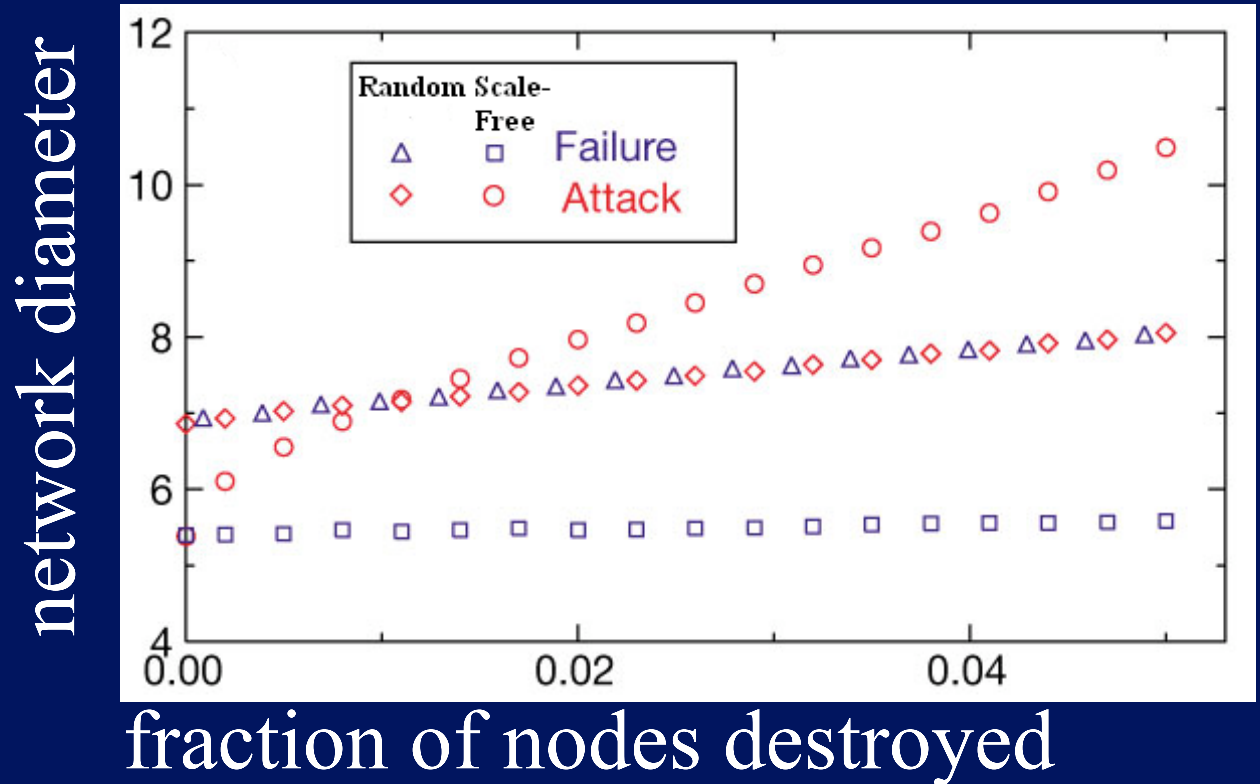


Power Law Distribution of Node Linkages



Internet Tolerance to Attacks and Failures

- Scale-free networks are failure-tolerant
- Random networks are attack-tolerant



Source: R. Albert, H. Jeong, and A.-L. Barabasi, "Error and attack tolerance of complex networks," Nature, vol. 406, no. 6794, 2000, pp. 378-82.

Ways to Deal with Attacks

- Service continuity
  - Same as for FT, plus
  - Heterogeneity
    - Diversification
      - Avoid monocultures
    - Randomization
      - Avoid “hubs”
- Disaster recovery
  - Same as for FT

Summary for Availability

- Availability in the presence of failures
  - FT terminology
  - k fault tolerance
  - two army problem
  - Byzantine Generals problem
  - Services continuity and disaster recovery
- Availability in the presence of attacks
  - Failures vs. attacks
  - Random vs. scale-free networks
  - Internet tolerance to attacks and failures
  - Services continuity and disaster recovery