Survivable Real-Time Network Services

DARPA Next Generation Internet Meeting
26-29 October 1998

David L. Mills
University of Delaware
mills@udel.edu

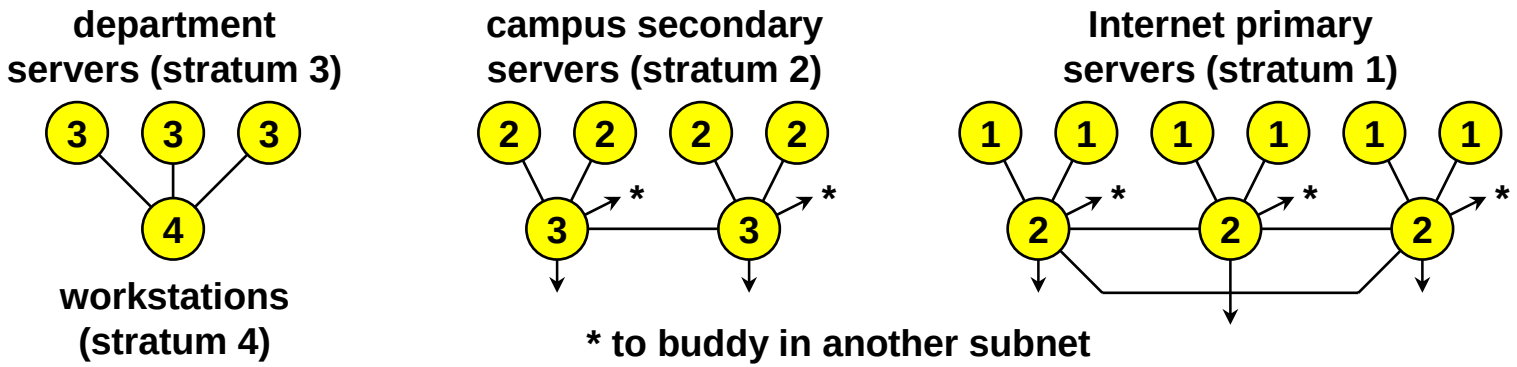
HTML, PostScript and PowerPoint versions of this presentation are available at http://www.eecis.udel.edu/~mills

Sir John Tenniel; *Alice's Adventures in Wonderland,* Lewis Carroll

NTP autonomous system model

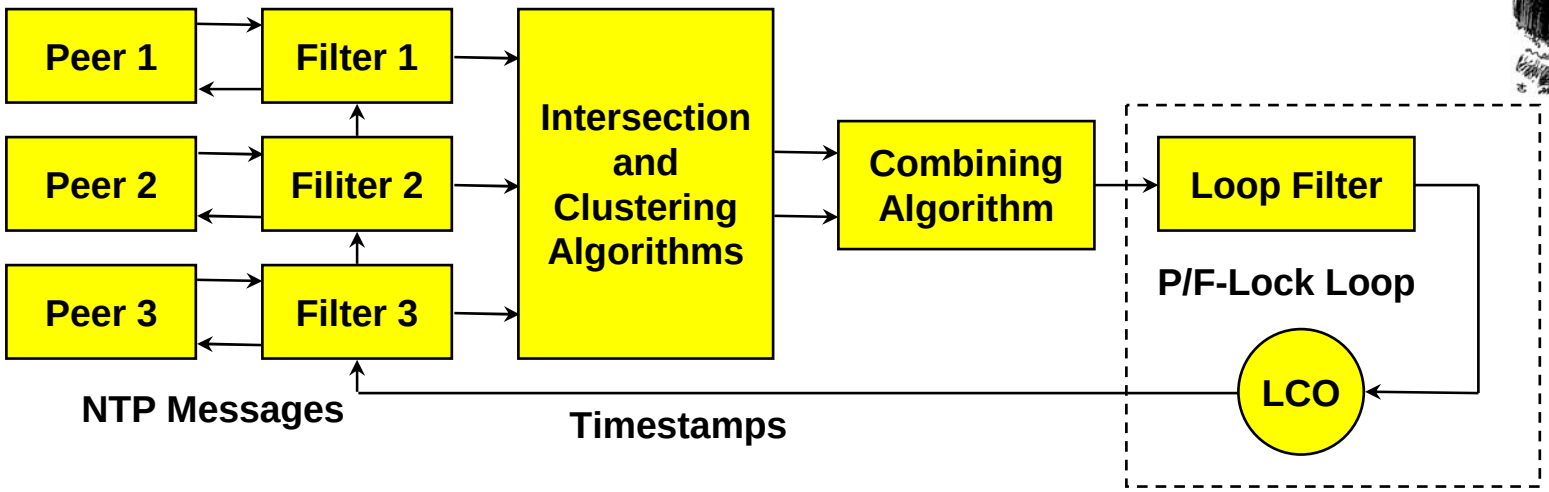
- Fire-and-forget software
  - Single software distribution can be compiled and installed automatically on most host architectures and operating systems
  - Run-time configuration can be automatically determined and maintained in response to changing network topology and server availability
- Autonomous configuration (autoconfigure)
  - Survey nearby network environment to construct a list of suitable servers
  - Select best servers from among the list using a defined metric
  - Reconfigure the NTP subnet for best accuracy with overhead constraints
  - Periodically refresh the list in order to adapt to changing topology
- Autonomous authentication (autokey)
  - For each new server found, fetch its cryptographic credentials from public databases
  - Authenticate each NTP message received as sent by that server and no other
  - Regenerate keys in a timely manner to avoid compromise

The NTP subnet



- NTP synchronizes the clocks of hosts and routers in the Internet
- Time synchronization flows from primary servers synchronized via radio and satellite over hierarchical subnet to other servers and clients
- NTP provides submillisecond accuracy on LANs, low tens of milliseconds on typical WANs spanning the country
- NTP software daemon has been ported to almost every workstation and server platform available today, including Unix, Windows and VMS
- Well over 100,000 NTP clients and servers are now deployed in the Internet and its tributaries all over the world

How NTP works



- Multiple servers/peers provide redundancy and diversity
- Clock filters select best from a window of eight clock offset samples
- Intersection and clustering algorithms pick best subset of peers and discard outlyers
- Combining algorithm computes weighted average of offsets for best accuracy
- Loop filter and local clock oscillator (LCO) implement hybrid phase/frequency-lock (P/F) feedback loop to minimize jitter and wander

Goals and non-goals

- Goals
  - Robustness to many and varied kinds of failures, including Byzantine, fail-stop, malicious attacks and implementation bugs
  - Maximum utilization of Internet multicast services and protocols
  - Depend only on public values and certificates stored in secure directory services
  - Fast operation using a combination of public-key and private-key cryptography
- Non-goals
  - Administrative restrictions (multicast group membership control)
  - Access control - this is provided by firewalls and address filtering
  - Privacy - all protocol values, including time values, are public
  - Protection against out of order or duplicated messages - this is provided by the NTP protocol
  - Non-repudiation - this can be provided by a layered protocol if necessary

Autonomous configuration and authentication - issues

- Configuration and authentication and synchronization are inseparable
- Autonomous configuration (autoconfigure)
  - Centralized configuration management does not scale to large networks
  - Finding optimal topologies in large subnet graphs under degree and distance constraints is NP-hard
  - Greedy heuristics may not produce good topologies in acceptable time
  - Solution may involve span-limited, hierarchical multicast groups and add/drop heuristics
- Autonomous authentication (autokey)
  - Centralized key management does not scale to large networks
  - Symmetric-key cryptosystems require pairwise key agreement and persistent state in clients and servers
  - Servers cannot maintain persistent state for possibly thousands of clients
  - Public-key cryptosystems are too slow for good timekeeping
  - Solution may involve a combination of public and private key cryptosystems

Autonomous configuration - approach

- Dynamic peer discovery schemes
  - Primary discovery vehicle using NTP multicast and anycast modes
  - Augmented by DNS, web and service location protocols
  - Augmented by NTP subnet search using standard monitoring facilities
- Automatic optimal configuration
  - Distance metric designed to maximize accuracy and reliability
  - Constraints due to resource limitations and maximum distance
  - Complexity issues require intelligent heuristic
- Candidate optimization algorithms
  - Multicast with or without initial propagation delay calibration
  - Anycast mode with administratively and/or TTL delimited scope
  - Distributed, hierarchical, greedy add/drop heuristic
- Proof of concept based on simulation and implementation with NTP Version 4

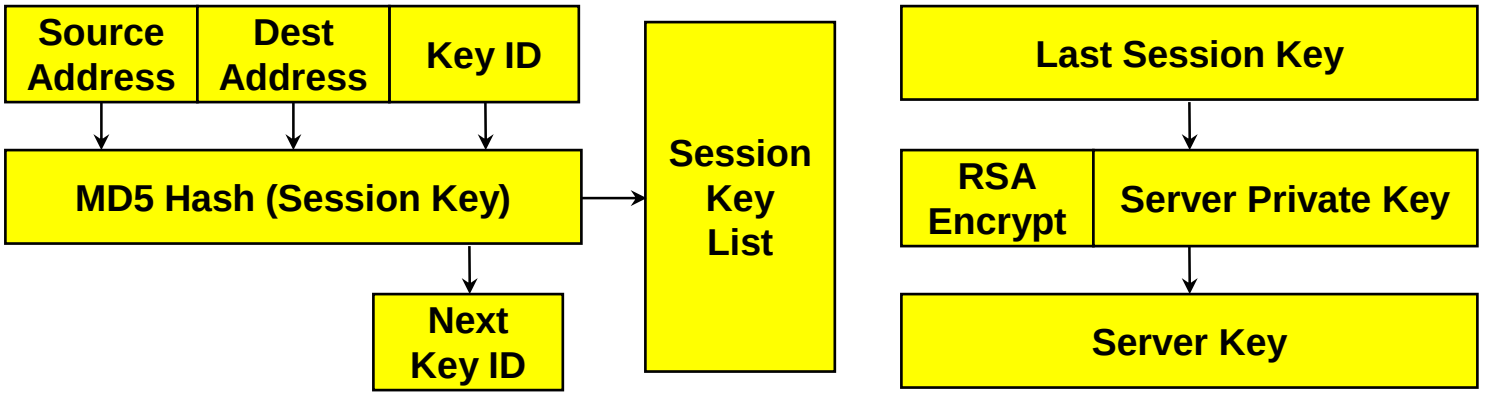
NTP configuration scheme

- Multicast scheme (moderate accuracy)
  - Servers flood local area with periodic multicast response messages
  - Clients use client/server unicast mode on initial contact to measure propagation delay, then continue in listen-only mode
- Manycast scheme (highest accuracy)
  - Initially, clients flood local area with a multicast request message
  - Servers respond with multicast response messages
  - Clients continue with servers as if in ordinary configured unicast client/server mode
- Both schemes require effective implosion/explosion controls
  - Expanding-ring search used with TTL and administrative scope
  - Excess network traffic avoided using multicast responses and rumor diffusion
  - Excess client/server population controlled using NTP clustering algorithm and timeout garbage collection

NTP authentication - approach

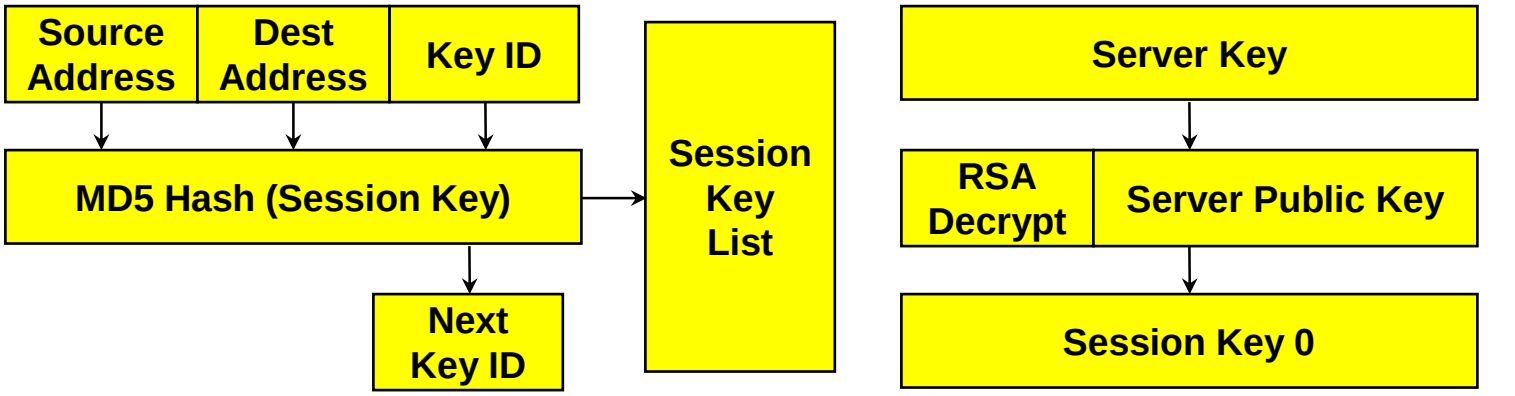
- Authentication and synchronization protocols work independently for each peer, with tentative outcomes confirmed only after both succeed
- Public keys and certificates are obtained and verified relatively infrequently using Secure DNS or equivalent
- Session keys are derived from public keys using fast algorithms
- Each NTP message is individually authenticated using session key and message digest (keyed MD5 or DES-CBC)
- NTP is run individually in unauthenticated mode for each peer to compute offset from system clock, together with related clock data
- If authentication data incomplete, clock data are marked tentative
- If the clock data incomplete, authentication data are marked tentative
- When both authentication and clock data are complete, the peer is admitted to the population used to synchronize the system clock

Generating the session key list



- Server rolls a random 32-bit seed as the initial key ID
- Server generates each session key as hash of IP addresses and key ID
- Low order 32 bits of the session key become the key ID for the next session key
- Server encrypts the last key using RSA and its private key to produce the server key
- Server uses the session key list in reverse order and generates a new one when exhausted

Using the session key list



- Server key and sequence number are included in the extension field of every NTP message
- Client generates each session key as hash of IP addresses and key ID
- Client verifies the low order 32 bits match the key ID of the most recent message
- If no match, a message may have been dropped, so the client hashes again, eventually to sequence number zero
- Server credentials are verified if the RSA decryption of the server key matches session key zero

Current progress and status

- NTP Version 4 architecture and algorithms
  - Backwards compatible with earlier versions
  - Improved local clock model implemented and tested
  - Multicast mode with propagation calibration implemented and tested
  - Distributed multicast mode protocol designed and documented
- Autonomous configuration *autoconfigure*
  - Distributed add/drop greedy heuristic designed and simulated
  - Span-limited, hierarchical multicast groups using NTP distributed mode and add/drop heuristics under study
- Autonomous authentication *autokey*
  - Ultimate security based on public-key infrastructure
  - Random keys used only once
  - Automatic key generation and distribution
  - Implemented and under test in NTP Version 4

Related Work

- Simulation of very large networks
  - Goal is to investigate global behavior and routing stability
  - Motivated by prior experience with unexplained instabilities in NSF Backbone network 1986-1988
  - Current simulator supports up to 3000 nodes now, eventually 10,000 nodes
  - Random network generated using Waxman model
  - Baseline Bellman-Ford routing algorithm now, eventually others
- Nanosecond Unix kernel
  - Goal is to improve accuracy from microseconds to nanoseconds in fast workstations and routers
  - Revised kernel modifications in SunOS, Solaris and Digital Unix
  - Currently working with NTP Version 4 on Sun IPC with cesium clock with 1000-ns resolution
  - Plan modified Alpha kernel clock with 3-ns resolution

Future plans

- Complete *autoconfigure* and *autokey* implementation in NTP Version 4
- Deploy, test and evaluate NTP Version 4 daemon in CAIRN testbed, then at friendly sites in the US, Europe and Asia
- Revise the NTP formal specification and launch on standards track
- Participate in deployment strategies with NIST, USNO, others
- Prosecute standards agendae in IETF, ANSI, ITU, POSIX
- Develop scenarios for other applications such as web caching, DNS servers and other multicast services

Further information

- Network Time Protocol (NTP): *www.eecis.udel.edu/~ntp*
  - Current NTP Version 3 and 4 software and documentation
  - FAQ and links to other sources and interesting places
- David L. Mills: *www.eecis.udel.edu/~mills*
  - Papers, reports and memoranda in PostScript and PDF formats
  - Briefings in HTML, PostScript, PowerPoint and PDF formats
  - Collaboration resources hardware, software and documentation
  - Songs, photo galleries and after-dinner speech scripts
- FTP server *ftp.udel.edu* (`pub/ntp` directory)
  - Current NTP Version 3 and 4 software and documentation repository
  - Collaboration resources repository
- Related project descriptions and briefings
  - See “Current Research Project Descriptions and Briefings” at *www.eecis.udel.edu/~mills*