



Scalability

Protocol Design – S-38.3157



A typical design argument:

“This does not scale...”

- ▶ Why?
- ▶ With respect to what?
- ▶ Does it have to?



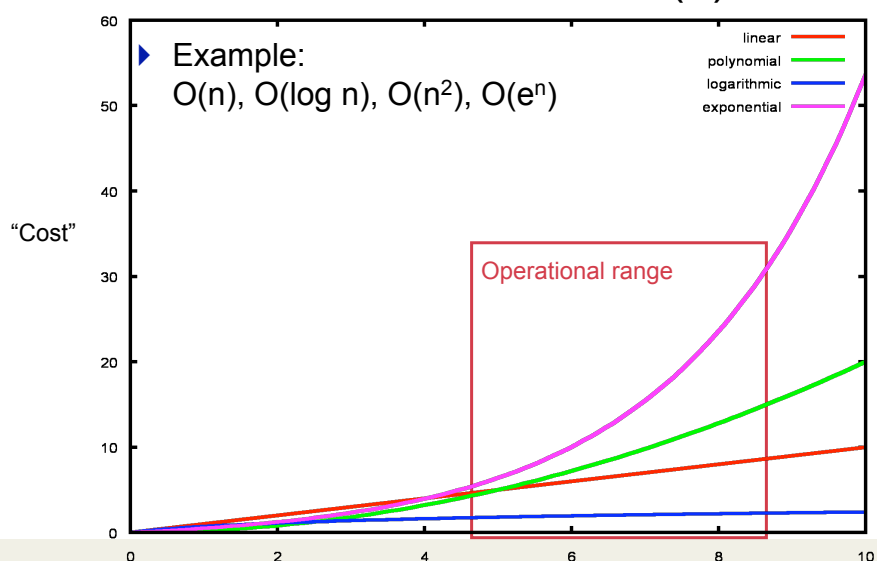
Scalability in General

Common use (not just) in communications

- ▶ Capability of a system to operate across a range of settings
 - As opposed to being constrained to a single operational point
- ▶ Measuring change / evolution of a system property
- ▶ Depending on a (set of) certain input parameter(s)
 - Applicability defined by the range of acceptable input parameters (for the which the resulting system properties are workable)
- ▶ Closely coupled to resource consumption (and thus fairness)
- ▶ Relation to complexity theory
 - Classification of resource consumption of algorithms depending on the input
Complexity classes (order of): $O(1)$, $O(n)$, $O(\log n)$, $O(n^k)$, $O(e^n)$

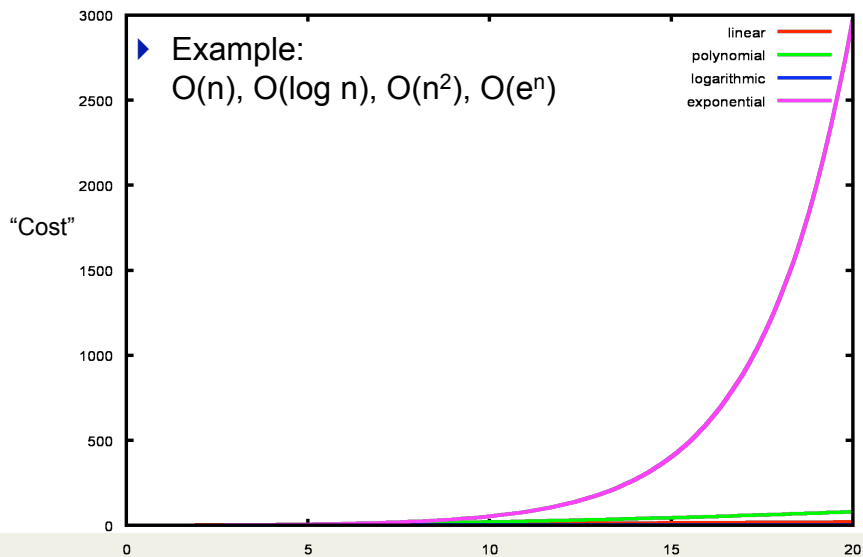


Scale as a Measure (1)





Scale as a Measure (2)



5



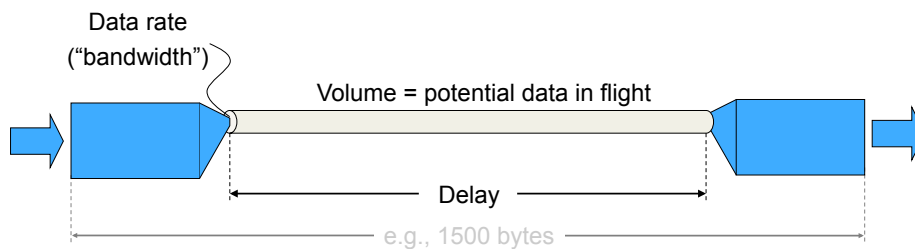
Areas of Scalability: Network Side

- ▶ Path length (number of hops, delay, delay variation)
 - Distance-dependent delay due to speed of light + processing/queuing delay per hop
 - Local link or same host vs. some 30 hops to Australia
 - < 1ms on a local link vs. several seconds via GPRS or satellite
 - vs. minutes or hours or days when talking to a spacecraft (or other remote peers)
 - Close to constant delay on a local link vs. several seconds jitter via satellite
 - Incurred by medium access protocol or in router queues due to other traffic
- ▶ Loss rate
 - Virtually no loss on a local wired link vs. <10% loss (typically 1-3%) for Internet traffic
 - Unpredictable loss rate and pattern for wireless networks
 - Individual losses (following some distribution) vs. bursty losses
- ▶ Data rate
 - Some 100 bit/s acoustic underwater modem vs. Tbit/s fiber optic link
- ▶ Degree of multiplexing
 - How much influence does the own traffic have on the network?
 - Access link vs. backbone link



Areas of Scalability: Network Side (2)

- ▶ Particular issue: long fat pipes (“bandwidth x delay” product)
 - Enabling efficient and quick full utilization without knowing pipe characteristics and third party traffic
 - No problem in traditional wired networks
 - Example: ISDN link @ 64 kbit/s x 10ms delay = ~800 Bytes

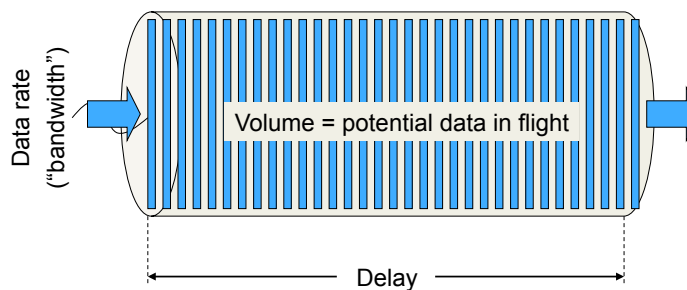


Only one packet in transit: first bits of packet are received before last bits have been sent



Areas of Scalability: Network Side (3)

- ▶ Long fat pipes (High “bandwidth x delay” product)
 - Many packets can be “in flight”



Examples: DVB-S2 geostationary satellite @ 90 Mbit/s x 250ms delay = ~2.8 MB
Fiber optic transatlantic link @ 10 Gbit/s x 25ms delay = ~31 MB



Areas of Scalability: Application Side

- ▶ Update or request rate
 - Measured in operations per second vs. per hour vs. per day vs. per year
 - Convergence time vs. period between two updates
 - Part of this: chattiness of the application protocol
- ▶ Item size
 - Data fitting into a single MTU or not
 - File size from some 10 bytes to 10 GB (and beyond)
 - Impact of per operation overhead
- ▶ Number of entities (users, networks, systems)
 - How many are active (sending operations)
 - How many must agree on common state as a result of the protocol operation
 - Dynamics: How does this number vary?
- ▶ Number spaces
 - In the protocol (see above) and in the operating system
 - Example: C10K problem: handling 10,000(s) clients with a server
 - Requires enough port numbers for demuxing, local identifiers (e.g., file descriptors), ...



Scalability Dynamics

- ▶ Network characteristics may vary heavily and frequently
 - Depending on a protocol entity's own activity
 - Depending on traffic generated by others
 - Depending on network routing changes (e.g., in response to failure)
- ▶ Application characteristics may vary
 - Size of data items (e.g., file size)
 - Number of involved systems interacting with one another (for group communications)
 - Number of involved systems operating in parallel (parallel clients for a server)
- ▶ Variations are usually not predictable
 - Example: Flash crowds



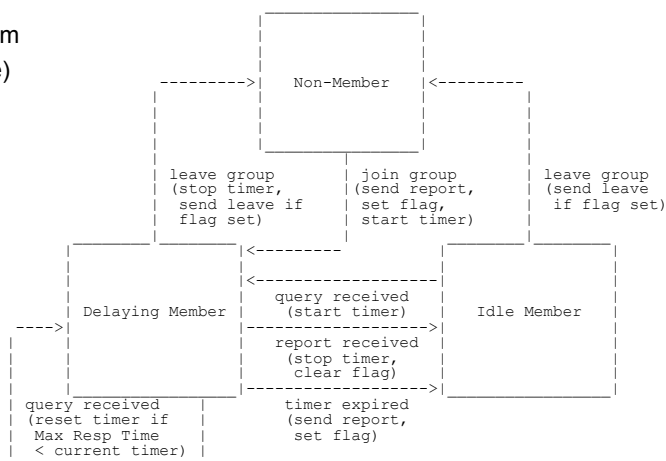
Meta Aspect: Complexity

- ▶ Protocol Complexity
 - MUST/SHOULD/MAY in the protocol spec, number of options
 - "Hard + easy = harder than hard"
 - State machine complexity
 - E.g., number of state and state transitions, synchronization requirements
 - # transitions (and interactions) to achieve a result, interdependence of entities
 - Operation complexity
 - E.g., parsing protocol messages
 - Computational complexity
 - E.g., crypto, routing, and lookup algorithms
 - ▶ Issue of backwards compatibility
 - Deployment considerations usually require dealing with older versions
 - Limits the freedom to introduce new functionality and better mechanisms
 - May lead to additional complexity if special treatment of "legacy nodes" is needed
- Evolvability



Example: IGMPv2 (1)

Plain IGMPv2 state diagram
for hosts (almost complete)





Implementation Scalability

▶ C10K Problem: Examples

- Frequency of interactions
 - Particularly expensive operations such as accepting and closing connections, security, ...
- Multiplexing and I/O handling
 - Processes vs. threads vs. single-threaded handling
 - Issues with system call efficiency (e.g., poll (), select ())
 - Solutions: kqueue (BSD) / epoll (Linux)
 - Processing many events simply takes time
- Data access
 - Seek operations on a hard drive when retrieving file blocks for many clients
 - Hard drives are "fast" unless multiplexed
 - Example: video-on-demand streaming
 - System bus bandwidth
 - I/O subsystem performance
 - Example: file transfer from/to a Windows machine (using cygwin)
 - Limited to 2–3 MB/s on 1.7 GHz laptop running MS Windows XP
- Interaction with other processes on the same machine



Implementation Scalability (2)

▶ Load balancing

- Use of server farms for load sharing
- Load distribution e.g. by means of DNS, proxies
- Possibly decentralized to improve access locality
 - And thus also avoid the impact of long paths
- Issue: need for synchronizing servers in a farm?



Operational Complexity

- ▶ Networks and systems need to be run
- ▶ How many parameters need to be configured?
 - How do they interact?
 - How much coordination (e.g., across different organizations) is needed?
 - (How) can misconfigurations be detected?
 - Manual vs. automated process
- ▶ Monitoring, diagnostics
 - Which parameters? Where? Frequency? ...?
- ▶ Failures
 - Graceful degradation vs. complete breakdown
 - How to track and debug failures?
 - How much action is needed for recovery?
 - How long does this take?



Meta Aspect: Economics

- ▶ Cost may be associated with data transmission
 - Rate, volume, packets, QoS, ...
- ▶ Cost is directly associated with implementation complexity
 - Manpower for system design, implementation, and testing
 - Device requirements
- ▶ Benefit is indirectly associated with protocol complexity
 - Successful deployments require working and interoperable products
 - Metcalfe's law
 - Complexity creates adoption hurdles
- ▶ Financial scaling
- ▶ (cf. Social scaling)



Limiting Scalability

- ▶ Scalability is usually another design tradeoff, per parameter
 - Scalability vs. protocol and implementation complexity, resource utilization, ...
 - Quick results vs. longer term perspectives
- ▶ Limiting applicability may be dangerous
 - Protocols may often be used outside their intended areas of application
 - Exceptions: e.g., intra-system communications in contained environments
- ▶ Dealing with scaling beyond expectation
 - Graceful degradation (of quality or functionality)
 - Clean failure
 - Make sure the protocol does not run havoc / create damage



Scalability Mechanisms by Example



Mechanisms: Timeouts

- ▶ Path length
 - Primary impact on delay and delay variation
 - Packet loss and degree of multiplexing covered below
- ▶ Issue: Protocols require timers and timeouts
 - Any statically selected value is likely to be wrong in some environment
 - Limiting factor for efficiency: too large ones may keep the network idle
 - Cause of unnecessary overhead: too small ones may lead to early retransmissions
 - Example: NFS used 500 ms
- ▶ Solution: Adaptive timers
 - Measure observed RTT and adjust timers accordingly
 - Use moving averages to avoid oscillation to short-term changes
 - Take a sufficiently conservative initial values that will not cause harm

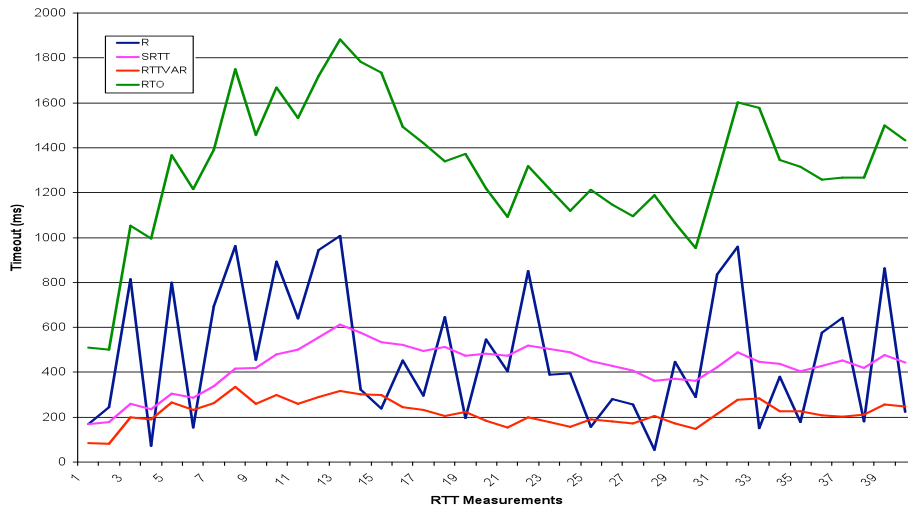


Example: TCP RTO Calculation

- ▶ TCP Retransmission Timeout (RTO)
 - Used to determine that a packet got lost and needs retransmission
 - Typically an indication of network congestion
 - Important implication: return to slow start operation
 - Underestimating worse than overestimating
 - Premature RTO will harm performance seriously (spurious retransmits, slow start!)
 - Late RTO will delay repair and hence also harm performance
- ▶ Algorithm
 - RTTVAR (RTT variation) and SRTT (smoothed RTT); G: clock granularity
 - Initial RTO = 3s
 - Upon first RTT measurement (R)
 - $SRTT = R$ $RTTVAR = R/2$
 - $RTO = SRTT + \max(G, K \times RTTVAR)$ [K=4]
 - For each subsequent RTT measurement (R')
 - $RTTVAR = (1 - \beta) \times RTTVAR + \beta \times |SRTT - R'|$ [$\beta = 1/4$]
 - $SRTT = (1 - \alpha) \times SRTT + \alpha \times R'$ [$\alpha = 1/8$]
 - $RTO = SRTT + \max(G, K \times RTTVAR)$ [K=4]



TCP RTO Example



Packet loss rate

- ▶ Loss rate
 - Example: Go-Back-N vs. SACK
 - Partially dependent on the degree of multiplexing
- ▶ Issue: distinguishing congestion losses and corruption losses
 - E.g., observing RTO as one hint for congestion likelihood
- ▶ General: congestion control
 - Reduction of data rate
 - Reduction of packet frequency
- ▶ Losses due to bit errors
 - Forward error correction (bit or packet-based)
 - Adaptive retransmission schemes

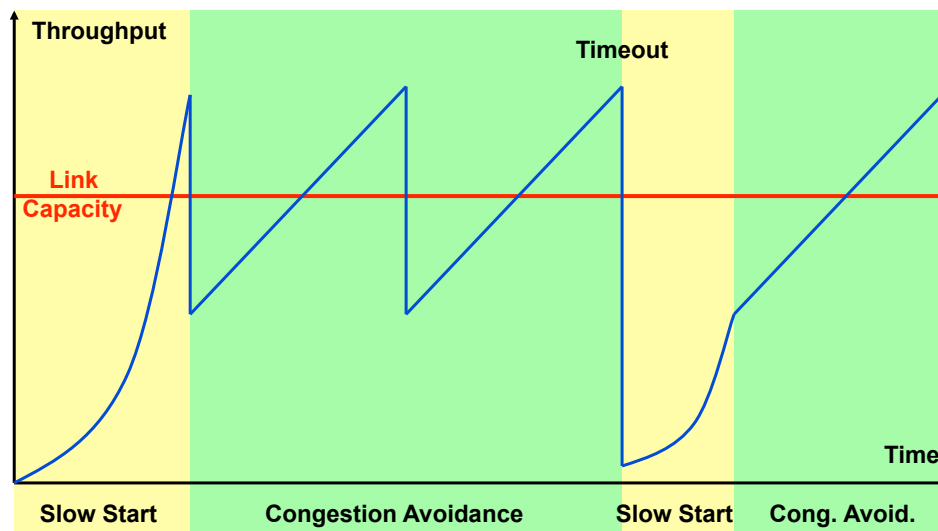


Mechanisms: Congestion Control

- ▶ Data rate
 - Obviously bounded by the slowest link in the path (upper bound)
 - Dependent on the current network load (and thus variable)
 - Other factors: delay and packet losses
- ▶ Issue: fair resource sharing vs. maximizing resource utilization
 - Protocol entities operate in unknown and changing environments
 - Again, no initial value for a data rate can be assumed
 - Pessimistic assumptions (low rate) may result in underutilization
 - Optimistic assumptions (high rate) may result in overload and lead to congestion
- ▶ Solution: dynamic adaptation of data rate
 - Many different options for rate adaptation
 - Not too conservative (to avoid wasting resources)
 - Not too aggressive (to avoid congestion)



Example: Simplified TCP Operation





Dealing with Long Fat Pipes

- ▶ Networks with large **delay x bandwidth** product
 - Sufficient bandwidth available
 - Yet limited communication performance
- ▶ Issue: peers cannot utilize available capacity
 - Limitations due to protocol parameters
 - Example: TCP Window size limited the amount of data in flight to 64 KB
 - Limitations due to protocol interactions
 - Examples: Lock-step operation of SMTP initial handshake, of HTTP when downloading web pages, of POP3 when downloading emails, of SMB when accessing files
- ▶ Some solutions
 - Sufficiently dimensioned parameters (expect the unexpected)
 - TCP: Window scaling option to multiply advertised window size by 2^x
 - Minimize the number of end-to-end interactions
 - SMTP, HTTP: pipelining of requests to avoid RTT penalty

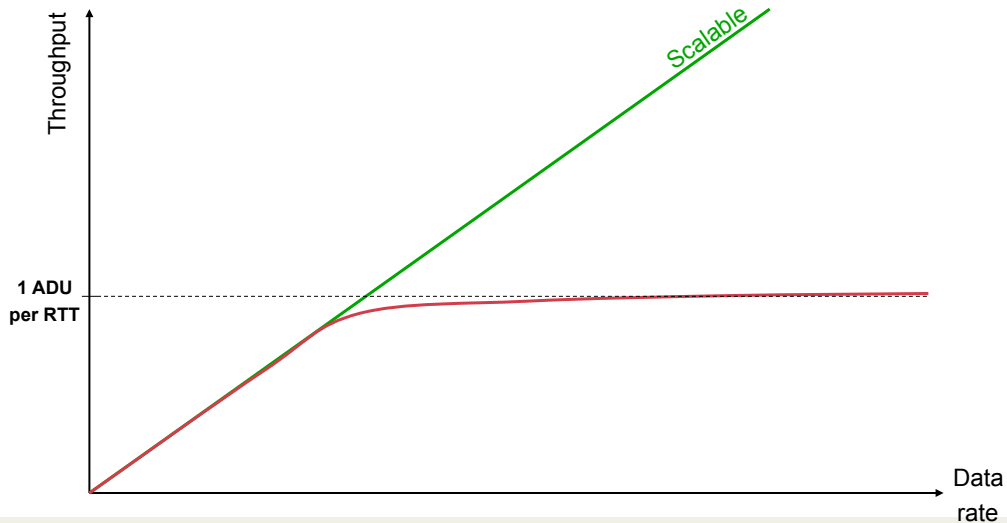


Real-world Examples: X11 and SMB in LFNs

- ▶ X11 Protocol
 - Designed for LANs
 - Frequent request-response interaction between client and server
 - Often lock step operation required: operation $n+1$ depends on result of operation n
 - Many small successive operations cause poor performance
 - Link capacity is not the bottleneck
- ▶ Server Message Block (SMB)
 - Resource (e.g., file and printer) access in LANs
 - RPC-style abstraction leads to horrible implementations
 - Synchronous function calls, apparently assumed to complete in virtually no time
 - Repeated invocation of the same methods
 - Extremely poor performance over long delay links (e.g., satellites)
 - Example: complete file transfer (~15 MB) takes 2.5s in LAN @ RTT=1ms but requires ~6 minutes @ RTT=1s
- ▶ General solution: Performance Enhancing Proxies (PEPs) for applications

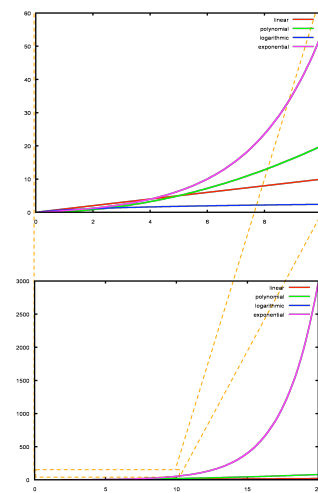


Lock-step Protocols hit an RTT Ceiling



Thursday Summary

- ▶ Scalability: constant, logarithmic, linear, polynomial, exponential relationships
- ▶ Relative to multiple parameters: (sizes, rates, numbers, ...)
- ▶ “Protocol designer’s toolkit”
 - Adaptiveness (measure and react)
 - Decentralization, caching, ...
- ▶ Scalability also is about economics
 - Throw more hardware vs. better design





Impact of Data Rate

- ▶ Complexity of protocol and algorithmic operations may be limited by data rate
 - Different scaling of data rate and processing power
 - Processing is one potential bottleneck
 - Applies particularly to routers (but may also affect endpoints)
- ▶ Examples
 - Plain packet forwarding vs. policy-based routing
 - The former works across all wire speeds ("fast path processing")
 - The latter is limited to "slower" links (no per packet route calculation possible)
 - Tradeoff across different crypto algorithms
 - Public key cryptography operations expensive to compute: limited to small amounts of data and occasional use
 - Symmetric crypto algorithms suitable for higher data rates



Scaling to large Numbers (1)

- ▶ ...of items, users, systems
- ▶ Decentralized operation
 - Also helps with load sharing for implementations
- ▶ Passive: Caching
 - Web caches, DNS
- ▶ Active: Content replication
 - Streaming servers for media on demand content
 - Web servers for news agencies
- ▶ Tradeoff: keeping content current (and synchronized)
 - How well can applications deal with (slightly) out-of-date data?
 - Update frequency from server side only (web and streaming servers)
 - May incur significant complexity with update operations from client (e.g., distributed databases)



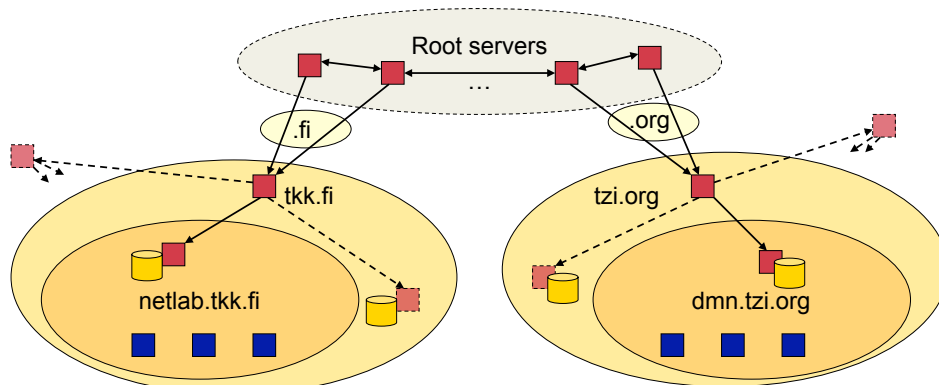
Example: DNS

- ▶ Key features
 - Distributed model: cooperation between servers
 - Redundant servers: avoid single point of failure in zone
 - one primary and one or more secondary servers
- ▶ Hierarchical structure of domain names
 - For decentralized administration and operation
 - Straightforward delegation of responsibility
- ▶ General purpose: not restricted to IP addresses
 - Could map anything
 - Stores additional information about domains
- ▶ Scalability mechanisms
 - Bottom-up search: exploit locality of requests
 - Efficiency through caching
 - Active replication of partial information (DNS servers for domains vs. contents)



Example: DNS (2)

- ▶ Decentralization, delegation, locality



- ▶ Dynamic Delegation Discovery System (DDDS)



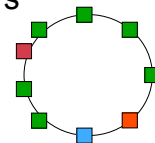
Limits to Caching

- ▶ Diversity of item access must be limited relative to cache size
 - Working set must fit
 - Locality helps
- ▶ Sufficient number of requests relative to lifetime of cached items
- ▶ Example: Route caching
 - Works for AS router with small number of entries in forwarding information base
 - Does not work for backbone router with large FIB



Example: DHTs (1)

- ▶ Information storage no longer relies on dedicated servers
 - Removes the load on a small number of entities
 - Distributes it across (more or less) equal peers
 - Often used as a synonym for scalability
- ▶ Simplified Operation
 - Peers are organized according to some topology (ring, space, cube, ...)
 - Nodes know their neighbors and probably other nodes
 - (Information about) resources are stored on one or more of these peers
 - An index for each resource (=the hash) points (in)directly to their location(s)
 - This hash table is maintained distributed across all nodes
 - If a request is made to a node
 - The request "string" is hashed (algorithmically) and yields a node storing the resource
 - The request is forwarded along the topology (following known links) to the target node
 - The request is answered directly by this node





Example: DHTs (2)

- ▶ Some thoughts on the scalability of DHTs
 - First of all: many variants exist for good reasons: design tradeoffs
- ▶ Topology maintenance
 - Maintaining an overlay requires effort as nodes join and leave
 - Adjusting pointers to topological neighbors (and more remote nodes)
 - Moving/replicating resources stored on a leaving node to others
 - Populating a newly arriving node
 - Essentially: balancing the topology and ensuring that the hashing works
 - Unlike servers, clients may get disconnected, turned off, are unreliable, etc.
 - Introduction of “super-peers” (which are well connected and appear more reliable)
 - Possibly a dynamic process
- ▶ Lookup/update overhead
 - Information resources get stored, modified, and retrieved
 - Extreme 1: Make the lookup efficient and the update expensive
 - Extreme 2: Make the lookup expensive but the update cheap
 - Usually something in-between, possibly augmented by some caching



Scaling to Large Numbers (2): Deferring Operations

- ▶ Late binding
 - Resolve bindings (e.g. name to address mapping) as late as possible
 - Allows operation with partial knowledge – no global synchronization needed
 - Example: IP telephony
 - SIP User Agents may defer address resolution to their local server
 - Saves complexity in the endpoint
 - Saves communication overhead (as the server may perform efficient caching)
 - User location is only performed at the called user’s server
 - Counter-example: DNS
 - IP address needed for any communication
 - Name to address resolution carried out by the endpoint
 - Requires globally connected naming infrastructure
 - Exception (counter-counter-example): HTTP proxies resolve names for their clients
- ▶ Lazy Evaluation
 - Do not calculate a result before it is known to be really needed



Scaling to Large Numbers (3): Hierarchies

- ▶ “Divide and conquer”
 - Subdivide the large problem into more manageable pieces
- ▶ Example: Directory services
 - E.g., DNS hierarchy
- ▶ Example: Routing protocols
 - Autonomous systems
 - Distinction between Inter-domain and intra-domain routing
 - Only network prefixes are known outside a domain, internal structure is “hidden”
 - Network prefixes may be further aggregated
 - CIDR: Classless inter-domain routing
 - Aggregation of class C addresses
 - IP forwarding operation
 - Across networks based upon IP addresses
 - Within networks based upon link layer addresses



Example: Routing Information Protocol (RIP)

- ▶ Protocol-inherent problems for Distance Vector routing
 - Limit the applicability to larger networks
- ▶ Datagram-based route reporting
 - # items to report vs. MTU size
 - Incremental reporting: all routes need to be sent once every 30s
 - Impact on convergence: impossible to tell the absence of an entry
- ▶ Bandwidth requirements for updating
- ▶ Instability during routing changes
 - Convergence to a new consistent view of the network takes a while
 - Temporary path unavailability or loops observable from the endpoints
- ▶ Counting to infinity
 - Need to define infinity so that converging does not take too long
 - Choice: 16!
 - Hard limit on network diameter



Scaling to Large Numbers (4): Degradation

- ▶ Accuracy requirements may change depending on the number of involved entities
 - Phone conversation vs. small group discussion vs. lecture vs. concert
 - But: distributed database transactions
- ▶ Possible tradeoffs
 - Completeness: not all information (state) may need to be known
 - Select representatives
 - Allow for incomplete views
 - Timeliness: state changes may not need to be communicated immediately
 - Allow temporary inaccuracies
 - Functionality: not all operations make sense for all group sizes
 - Example: Repairing packet losses in a TV broadcast with 1 M receivers vs. repairing packet losses in a three party conference
- ▶ “Loose coupling”



Example: RTCP

- ▶ Provides group membership and reception quality information for RTP sessions
- ▶ Must scale with the number of group members
 - Must not take up too much network capacity (rate-limited!)
- ▶ Overall “RTP session bandwidth”
- ▶ Default: 5% of the session bandwidth for RTCP
 - Takes role (sender or receiver) into account
 - Up to 25% of session members are senders: 3.75% for receivers, 1.25% for senders
 - More than 25% of session members are senders: share data rate proportionally
- ▶ Scalable RTCP transmission interval
 - Based upon the group size, RTCP data rate, average RTCP packet size
 - Independently observed by all receivers → calculate their own rate
 - Randomization to avoid synchronization over time
 - Timer reconsideration in case many nodes enter or leave in parallel
 - Default minimum: 5s (2.5 seconds for initial packet)



Delegation and Roles

- ▶ Distinct roles with different responsibilities for protocol entities
 - Motivated by system/network design, efficient protocol operation, robustness
 - Explicitly assigned (by configuration) vs. self-organizing (inside the protocol)
- ▶ Supports division of tasks and helps limiting complexity
- ▶ Examples
 - OSPF: Designated Router and Backup Designated Router for broadcast nets
 - Only one router is responsible for forwarding packets
 - Similar concepts for multicasting
 - IGMP: Designated Querier for IGMP membership polling
 - Peer-to-Peer systems: supernodes vs. regular nodes
 - Reliable Multicasting: Repair heads, DLRs, Repetitors for local repair (packet retransmission) in subgroups or subtrees
 - Multicast Congestion Control: Current Limiting Receiver (CLR) or Selected ACKer to determine acceptable transmission rate



Scaling to Small and Large Numbers (5)

- ▶ Careful choice of field sizes and constants required
 - Avoid fixing if possible
 - If necessary, use foresight (tradeoff: overhead vs. longevity)
 - Attention: variable length fields increase processing complexity
- ▶ Examples for limiting field sizes and structure
 - 32 bit IPv4 address and its initial (wasteful) address classes
 - IPv4 option space
 - TCP window size (see above)
 - TCP header extension space
 - Port number size (16 bits) and the range allocation (only 16 K dynamic ports)
- ▶ Examples for constants
 - 16 = infinity in RIP
 - Initial RTCP timer, minimal RTCP interval



Concluding Remarks

- ▶ Scalability means adaptivity
 - “Optimization” problem for multiple input and output parameters
 - Adaptation works only in the order of RTT
 - Beware of oscillation
 - Tradeoff in various dimensions
 - But: Don’t let your protocol get too complex
 - Must be implemented after all
- ▶ Scale as you need!
 - But be aware of your requirements
- ▶ When you don’t know what to expect: be conservative
- ▶ Remember: your protocol might be successful!