Thinking Different

Protocol Design

Assumptions about Operating Environments

- We always make assumptions about operating environments
- These obviously do not hold everywhere
  - Wireless communications
  - Node mobility
  - Size, processing power, and energy constraints
  - Persistence of available communication links
- Special application areas may require different protocol designs
  - Stronger vertical integration, heavy tailoring, less reusability, closed env.
- Three case studies (out of many…)
  - The Onion Router (TOR)
  - Sensor networks
  - Delay-tolerant networking

Anonymity in the Internet: The Onion Router (TOR)

More information: http://tor.eff.org

The Desire for Anonymity

Internet Users may want to stay anonymous:

- With respect to providers of services
  - To avoid excessive data collection
    - Cf. cookie debate
    - What does a monster.com spike from company X employees tell you?
  - To circumvent country restrictions
  - To conceal competitive analysis
- With respect to unknown adversaries
  - Protect customers from [visited] ISP ("peeking is irresistible")
  - Protect victim from criminal attacker
    - Kids from stalkers, anyone from blackmailers, traveler from hostage takers, …
  - Protect anyone from secret services (corrupt ones, those of other countries)
  - Protect citizen from oppressive government

But, Criminals also want Anonymity!?

- Yes.
- Actually, they like it so much, they already have it.
  Many options are available to criminals:
  - Forged ID
  - Identity theft
  - Stolen cellphones
  - Botnets, spyware, viruses, …
- Not providing an anonymity service is unlikely to stop crime
- If anonymity is outlawed, only outlaws will have anonymity

What is Anonymity

- Your actions cannot be traced back to you
  - Inverse of Accountability
- They may still be traced back to your **anonymity set**
  - E.g., customers of a physical shop (paying cash) must have been in town
  - E.g., users protected by a specific anonymity service must have used that service
- Problem for network communication:
  What if I want to able to receive return communication?

Basic idea: Anonymizer

- Alice talks to Intermediary, Intermediary talks to Bob
  - Alice is effectively hidden behind Intermediary's anonymity set
- Problem: What if the Intermediary is subverted?
  - Post-communication: Perfect forward secrecy can help
  - Pre-communication: ———
- Refinement: Chaining anonymizers
  - Even if some are subverted, they only know previous and following node
  - Need to guard against majority attacks, though

Why isn't this a standard offering?

- Anonymity cannot be created by sender or receiver
  - E.g., nobody can run their own anonymizer alone for themselves!
  - Others need to produce traffic to cover an anonymous sender
- Usability, (reasonable) efficiency, reliability, cost become security objectives!
- Reluctance to provide infrastructure for others to use
  - And misuse
    - Anonymity implies misuse cannot be prevented by excluding perpetrator
  - Legal liability not yet tested in court
    - "Should be OK" not enough for many potential anonymity service operators
  - Attackers can weaken anonymity systems by relying on this reluctance
- **Deployability** becomes an overriding concern

Classical "high-latency" anonymizer: MIX

- MIX: Server that receives a mail message, decrypts it using a private key, and sends it on to next hop (in decrypted part)
  - Chain of MIXes protects against small number of subverted ones
  - Client only needs to know address and public key of a number of MIXes

Attack: correlate input and output

- To thwart traffic analysis by **time**: **delay** by a random time ("mix")
- To thwart traffic analysis by **size**:
  - Pad messages to constant size
  - Chop larger message into "packets", which are MIXed independently
    - Only "Exit MIX" reassembles
- Mixminion, http://mixminion.net/

The threat model

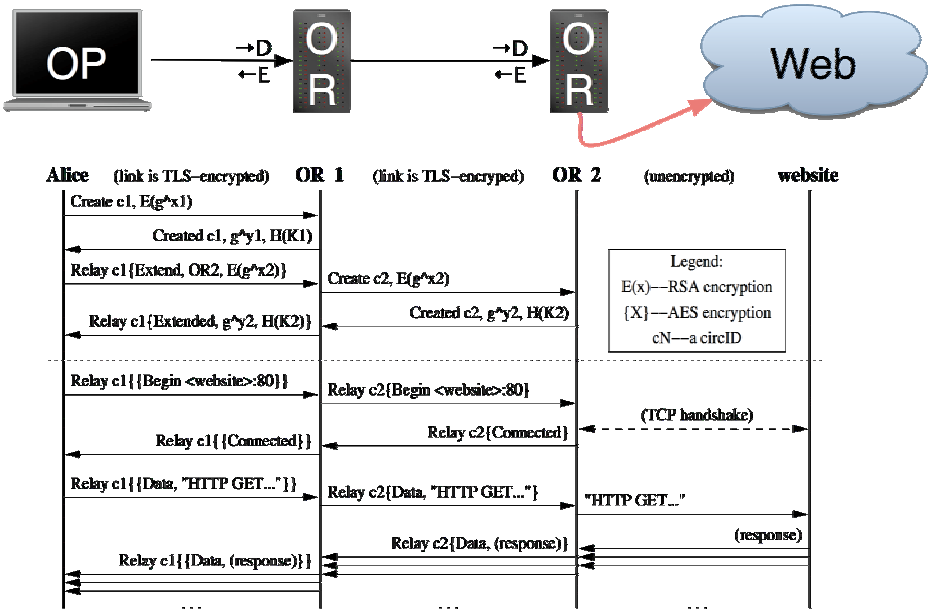
- Global passive adversary: attacker controls all your paths
- Traffic analysis: correlate your traffic with traffic on peer
  - Countermeasure: introduce (variable) delay (high, e.g., 2 days)
- Browsing, chat, SSH: need **low latency**
- Impractical to completely thwart traffic analysis
  - Particularly hard: "traffic confirmation": confirm suspected correlation
- Active attack:
  introduce timing pattern at one end and confirm it at other end
- Solution currently **impossible**

If you don’t like the answer, change the question!

- Give up:
  - Protection against **global** passive attacker
  - Protection against traffic **confirmation**
- Continue to protect against powerful attacker that can
  - observe some fraction of network traffic;
  - generate, modify, delete, or delay traffic;
  - operate anonymizers of his own;
  - compromise some fraction of the anonymizers.

The Onion Router (TOR)

- TOR addresses low-latency anonymity:
- Chain of anonymizers: “onion routers”
  - Selected by source (“onion proxy”, OP)
  - For each “circuit”, each OR knows only predecessor and successor
- Padding: all traffic is in 512-byte "cells"
  - make traffic analysis harder
- Cells are unwrapped (forward)/wrapped (reverse) at each OR
  - Integrity checked at the exit (against "tagging" attacks)



Perfect forward secrecy

- Telescoping: incremental circuit build from OP
  - Uninvolved ORs don't even know — cells are encrypted
- Use a fresh Diffie-Hellman for each new OR in the circuit
  - Once these keys are deleted: Perfect Forward Secrecy
  - Also helps with circuit build-up reliability
- Of course, exit OR does not provide PFS
  - But neither does the target system (website etc.)
  - Exit OR is enough "onion layers" remote from OP to provide good anonymity

Implementation issues

- Which layer?
  ➔ for TCP-based streams only
  - avoids need for kernel hacks (deployability!)
  - reduced timing sensitivity of traffic
  - IP packets reveal OS types and versions (OS fingerprinting)
  - exit policies would be much harder to define for IP packets
- Application integration: e.g., via SOCKS
  - Issue: DNS lookup
    - app calling gethostbyname reveals host to DNS server
  - Need socks4a/5 support in application, no gethostbyname calls
- Issue: “protocol cleaning” — not one of TOR’s jobs
  - E.g., use Privoxy to "clean" HTTP

Resource usage, fairness

- Rate limiting
  - OR operators can set a bandwidth limit
    - Token bucket approach
  - Make TOR deployment more attractive for potential operators
- Protocol multiplexing
  - TOR multiplexes TCP connections (circuits, streams)
  - window-based flow control (“congestion control”)
    - per-circuit and per-stream

Management

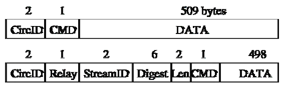
- Directory servers, downloadable (HTTP) OR list
  - Directory servers could also (anonymously) engage in testing ORs
- Exit policies:
  what traffic does an anonymizer allow to appear to be from it?
  - middleman (no exit)
  - private exit (talk to local hosts only -- increases security)
  - restricted exit (e.g., no port 25)
  - open exit
- **Variety in outcome**:
  TOR provides choices for OR operators
  - It would do deployment no good to try to enforce a single exit policy

Key Management, Rotation

- Key Management:
  - long-term key for TLS and signature of router descriptor
  - short-term onion key to negotiate ephemeral keys
    - rotated periodically and independently
- Circuits are considered for rotation every minute
  - are built in the background
  - Cannot immediately re-build circuit (destruction attack)

The TOR protocol

- Each OR maintains a TLS connection to every other OR
  - All communication in 512-byte Cells on these TLS connections
  - TLS provides hop-by-hop PFS and integrity protection
- Hop-by-hop Cell header:
  - 2-byte CircID (per TLS connection) + 1-byte command
  - Command can be: padding (NOP, also used for keep-alive), create/created, destroy
- Relay cell header: StreamID(2), Len(2), Cmd(1), Digest(6), Data(498)
  - Digest (6) -- first two bytes are zero (identifies exit/entry)
    - Implements leaky pipe scheme without hop-by-hop decapsulation
  - relay data
  - relay begin(IP/Name, port) ➔ connected (open stream)
  - relay end (close cleanly), or relay teardown (abort broken stream)
  - relay extend ➔ extended (telescoping); relay truncate ➔ truncated (untelescoping)
  - relay sendme (cc window open)
  - relay drop (NOP, long-range dummies)



Deployability

- The design must be deployed and used in the real world
- Thus it must not be expensive to run
  - (for example, by requiring more bandwidth than volunteers are willing to provide)
- Must not place a heavy liability burden on operators
  - (for example, by allowing attackers to implicate onion routers in illegal activities)
- Must not be difficult or expensive to implement
  - (for example, by requiring kernel patches, or separate proxies for every protocol)
- “Not covered by the patent that affected distribution and use of earlier versions”
- Cannot require non-anonymous parties (such as websites) to run TOR
- Client-side easily implementable on all common platforms
  - we cannot require users to change their operating system to be anonymous
  - currently runs on Win32, Linux, Solaris, BSD-style Unix, MacOS X, and probably others

Wireless Sensor Networks

Slide contributions by Dirk Kutscher (Uni Bremen TZI)

HELSINKI UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMMUNICATIONS AND NETWORKING

What is a Sensor Network?

- Term *sensor networks* describes an application class
  - Many different use cases and instantiations
  - Many different technologies
    - Network architectures, link layer technologies, routing protocols, application layer protocols etc.
- Wide range of characteristics
  - Fixed power supply vs. battery operation
  - Overall data rate
    - Maximum bit rate, always on vs. periodic suspension and activation
  - Number of nodes
    - Scalability
    - Network topology
  - Reconfigurability
    - Single-purpose vs. general-purpose

© 2008 Jörg Ott & Carsten Bormann

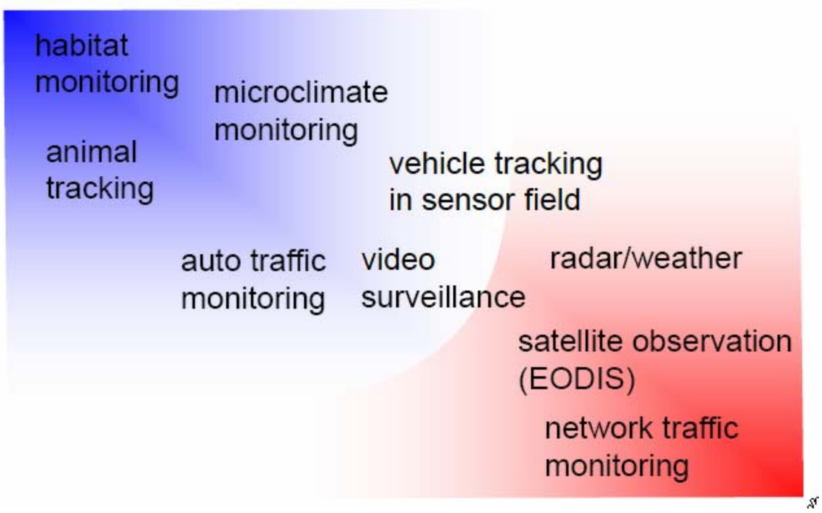
Sample Applications (1)

- Smart dust, e.g., chemical sensing
  - Many sensors (embedded systems), potentially large coverage areas
  - Power constraints
  - Robustness, tolerance for partial failures
  - Constant monitoring, constant data transmission
  - Low bit rate, "push" communications
  - May require automatic configuration, adaptation
  - May require ad hoc routing
  - May require specialized network design

Sample Applications (2)

- Wide area sensing networks, e.g., powered radar stations
  - Large geographic scale
  - Limited number of sensors, each node can be manually installed and configured
  - No power constraints
  - High data rates: 100 Mbps per node
  - Multiple consumers
  - Can be implemented with existing Internet based technologies
  - Requires additional technologies above IP
    - Content distribution, evaluation

Sample Applications (3)

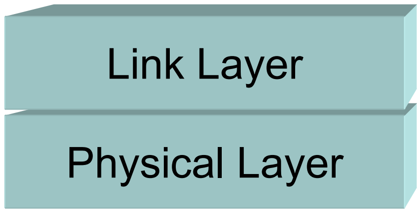


Protocol Design Issues: Physical Layer

- Wireless media
- Robust modulation
- Low power consumption
  - Adaptable transmission power

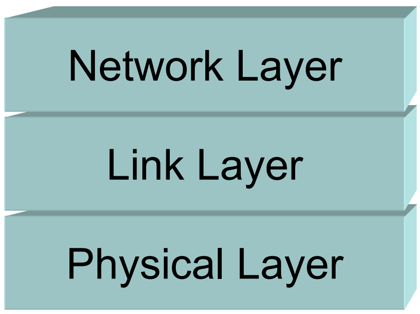
Protocol Design Issues: Link Layer

- Media access
- Power conservation
- Minimizing collisions
- Managing longer periods of inactivity
  - And synchronizing for transmission & reception
- Providing basic reliability



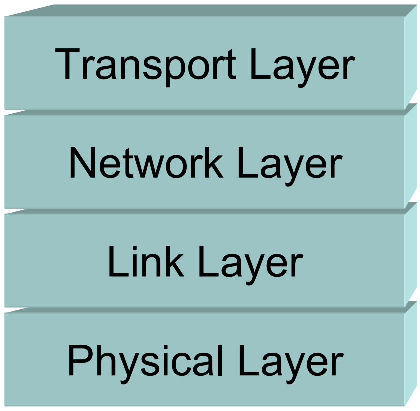
Protocol Design Issues: Network Layer

- Routing data between nodes
  - and to “sinks”, e.g., towards a data collector at the edge of a sensor field
- Self-organizing, self-healing
- Different requirements for addressing:
  - Atttribute-based, location-based, topology-based
- Point-to-point communication vs. group communication
- Internetworking with external networks



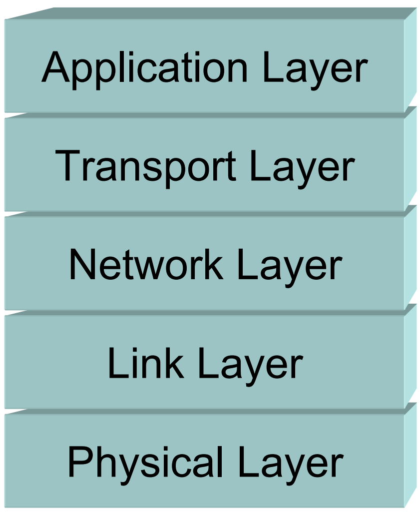
Protocol Design Issues: Transport Layer

- Transport protocols for
  - Controlling nodes
  - Coordinating sensor networks
  - Real-time transmission of sensor data
- Highly application-driven
  - Existing protocols not always appropriate
- Typically rather messaging-based than stream-based communication

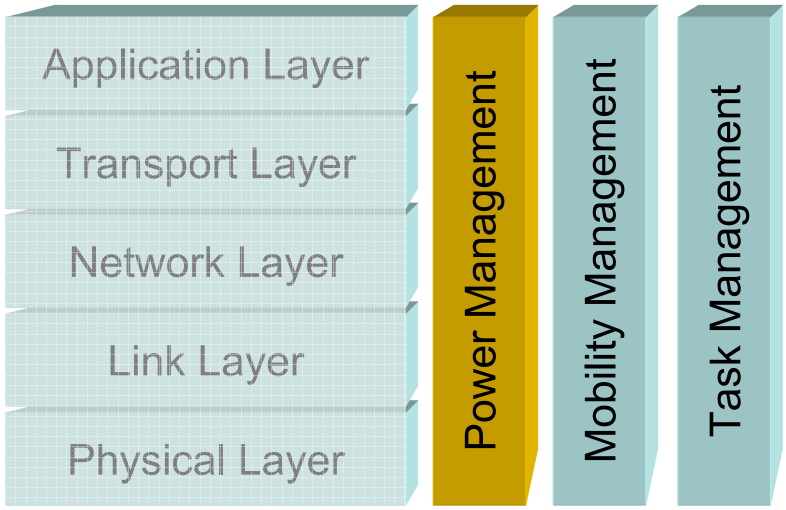


Protocol Design Issues: Application Layer

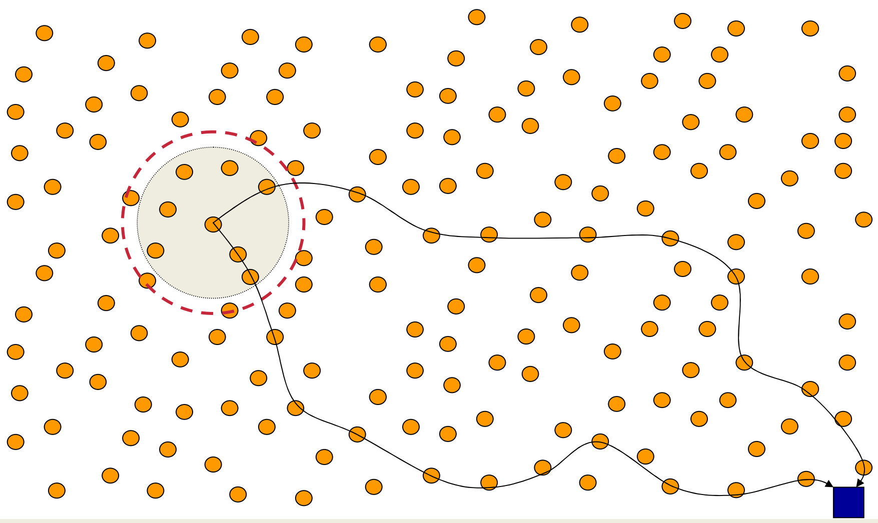
- Managing nodes of a sensor network
- Service location
- Data dissemination
- Different types of cooperation:
  - Sensor fusion
  - Real-time transmission
- Again, need to consider power-consumption



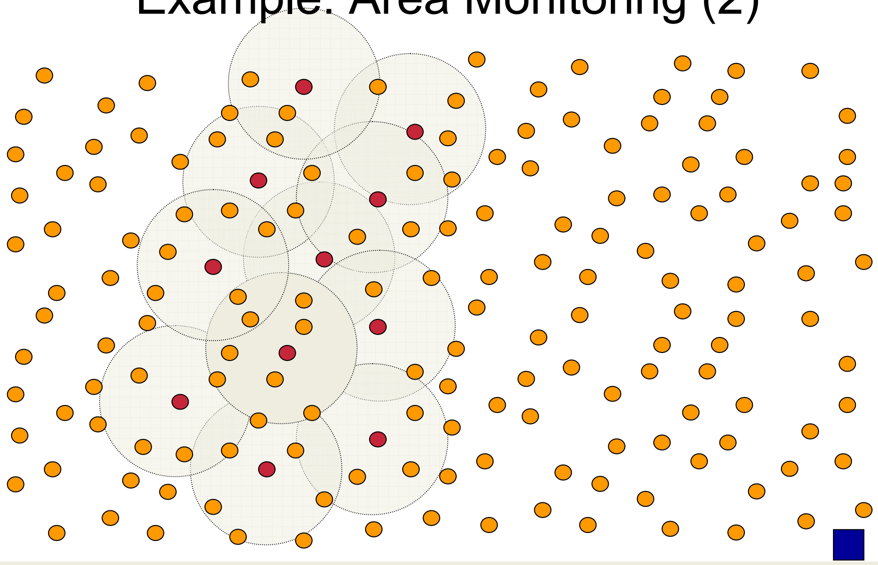
Cross-Layer Interaction



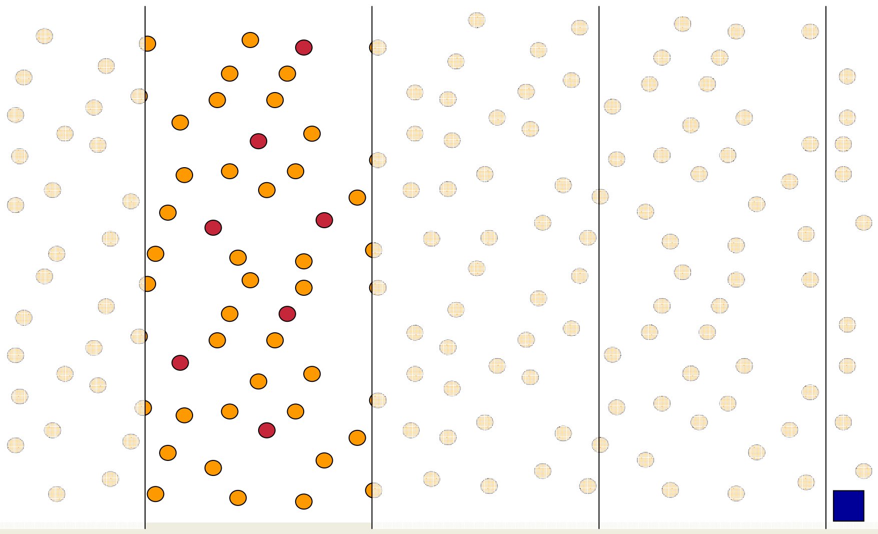
Example: Area Monitoring (1)



Example: Area Monitoring (2)



Example: Area Monitoring (3)



Summary

- Implementation of sensor networks highly application-driven
  - No single general-purpose solution
- Design influenced by extreme requirements
  - Power consumption, low complexity, cost per node
  - Applies to all layers
- Traditional protocol design strategies often not appropriate
  - Cross-layer interaction
  - Deviate from layered approach
  - Higher layer designed often influenced by characteristics of specialized physical and link layer protocols

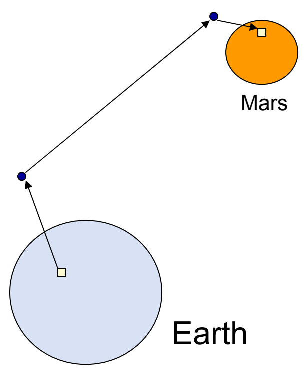
Delay-tolerant Networking (DTN)

Avoid (the Need for) Synchronous Communications

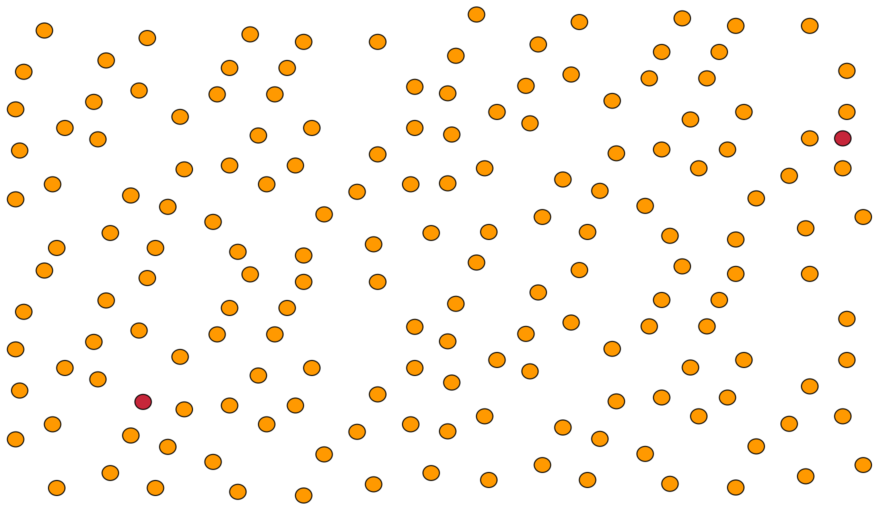
- Delays may be too long for interactive protocols
  - We have seen that RTTs in the order of seconds are already bad
  - How about RTTs or minutes or hours or even days?
- An end-to-end path to a peer may never exist
  - At least not at the order of time IP routers and end systems operate
- Delay tolerance implies disruption tolerance
  - If a peer, a link, or a path is currently not available, just wait until it comes back
  - Of hand the data to someone else who may have better chances of delivery
- Basic idea: follow asynchronous communication paradigm only
  - Simply modeled after email
  - Store and forward: wait for the next suitable opportunity to send
  - Store, carry, and forward: add physical data carriage as communication option
  - Realize end-to-end semantics where it belongs: at the application layer

Example 1: Deep Space Networks

- Communications with space crafts, space stations, satellites
  - E.g. Mars explorers
  - Low data rates, high error rate
  - Long propagation delays
    - Moon: ~3 seconds
    - Mars: ~2 minutes
    - Pluto: 5 hours
  - Link interruptions
    - Planetary dynamics
  - Scheduled communications
    - Pre-calculate next chance to communicate
    - Different requirements for “routing”
  - Retransmissions and interactive protocols are not workable

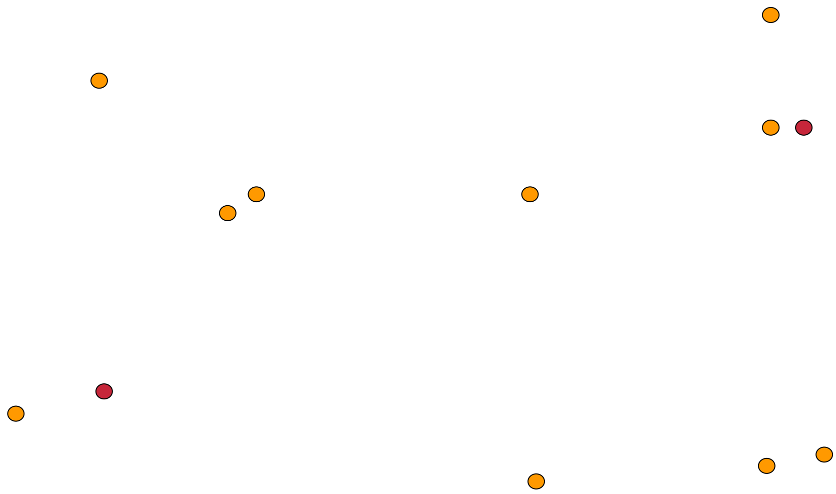


Example 2: Sparse Mobile Ad-hoc Networks

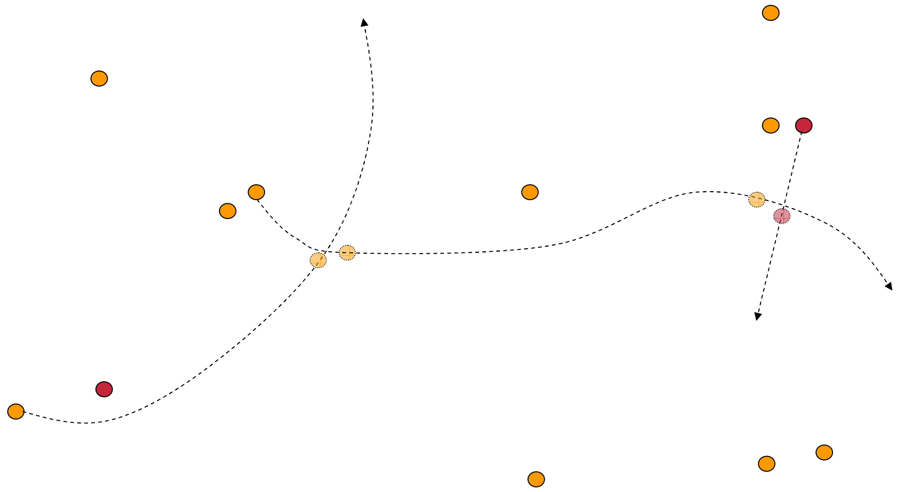


HELSINKI UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMMUNICATIONS AND NETWORKING

Example 2: Sparse Mobile Ad-hoc Networks



Example 2: Sparse Mobile Ad-hoc Networks



Example 3: Remote Internet Access

- Sámi Network Connectivity
  - Provide Internet Connectivity for Sámi population of Reindeer Herders
  - Nomadic users, no reliable communication facilities
  - Mix of fixed and mobile gateways
  - Routing based on probabilistic patterns of connectivity
  - E-Mail, Web-access, file transfer

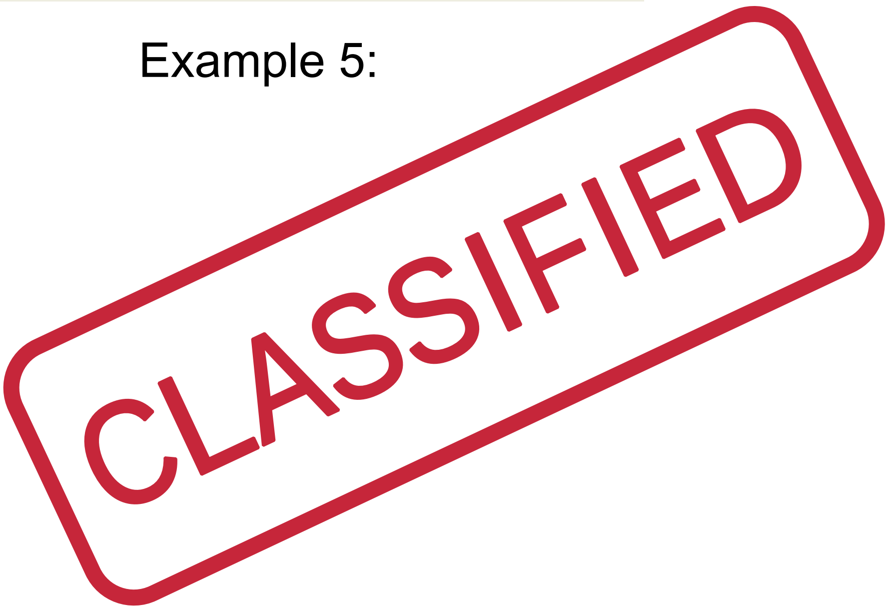
- DakNet
  - Internet access for remote villages in India and Cambodia
- Pocket-based communications
  - Exploiting people's motion for data transfer
  - Use buses, motor cycles, postal mail

Example 4: Acoustic Underwater Networks

- Interconnecting ocean bottom sensor nodes, autonomous underwater vehicles (AUVs), and surface stations (gateways)
  - Environment monitoring, underwater surveillance
- Propagation delay at the speed of sound (~1480m/s)
- Range and frequence significantly influence transmission loss
  - Doppler effects with moving vehicles
  - Multipath effects
  - Differences in deep and shallow water
- Range from 10s or meters to 1 – 10km, also 100 – 200km
- Data rates from 20 bit/s to a few kbit/s
  - Extremes: short range 500 kbit/s, long range 1 bit / minute
- Use “data buoys” for store and forward
  - Use ships for physical carriage (similar to “data mules” in sensor networks)

HELSINKI UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMMUNICATIONS AND NETWORKING

Example 5:

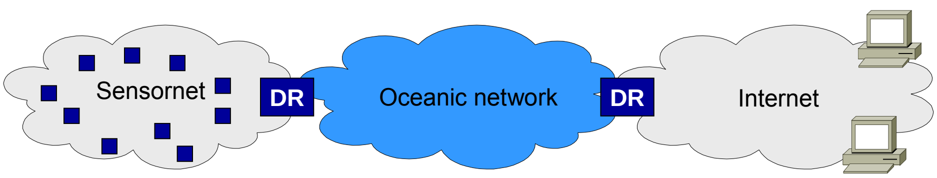


Delay-tolerant Networking (DTN)

- Following the paradigm of asynchronous communications
  - Often tailored to dedicated applications with specific protocols
  - But also suitable for some Internet "interaction": email, partly web, file transfer
  - Extreme variant: Postmanet
- Payload "units" of variable size
  - Ranging from a few bytes in sensor networks to typical IP packet size in some proposals to messages of virtually arbitrary size (again similar to email)
- New type of forwarding and routing: Store-and-(carry-and-)forward
  - A DTN-style router receives a unit and may take immediate action or delay it
  - Takes routing decision based upon known or potential paths
    - Present and future!
  - Forwards one or more copies of the unit when path becomes available

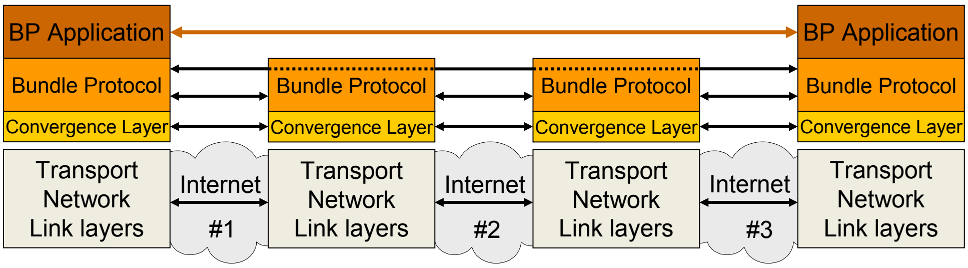
DTN RG Architecture (1)

- Delay-tolerant Networking Research Group in the IRTF
- Purpose: asynchronously interconnecting different internetworks
  - Which may be based upon arbitrary underlying technologies
  - Which may encompass just a link layer technology or a complete protocol suite
- Origin: deep-space communication (Interplanetary Internet, IPI)
  - How do entities in a long delay environment with intermittent connectivity talk?
- Example



DTN RG Architecture (2)

- **Bundle** as communication unit (like messages)
  - **Bundle layer** on top of underlying networks running **Bundle Protocol (BP)**
    - Implemented by **Bundle Protocol Agents** (aka hosts and routers)
  - Above the transport layer in the Internet (and similar architectures)
  - Above the link layer
- Mapping to lower layers defined by “convergence layer”



HELSINKI UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMMUNICATIONS AND NETWORKING

DTN Routing

- No longer “simple” connectivity graph as time dimension is added
  - Known present links (“contacts”)
  - Known future contacts
    - E.g., scheduled at a certain point in time
  - Potential future contacts
    - Peers are known but contact times are opportunistic
    - Peers are unknown and so are contact times
- New types of routing algorithms and “protocols”
  - Rarely based (up to now) on regular routing information exchange
    - Might be too expensive, always out of date, contact times too uncertain, etc.
  - Use of probabilistic routing instead
    - Simple 1: 1-hop routing: Wait until you meet your target (e.g., in MANETs)
    - Simple 2: flooding
    - Epidemic routing styles using history of contacts to determine future probability
    - Network coding and FEC-based distribution of data
    - Many variations presently under investigation
  - Evaluation metrics: delivery probability, delivery delay
- New challenge: congestion control of buffers in DTN routers

DTN RG Bundle Services and Protocols

- User services
  - Application registration ("bind ()")
    - Applications use URI-style scheme for identification
    - "Singleton" identifies a particular instance of an application
    - URIs may also refer to groups of receivers → Multicasting (interesting semantics!)
  - "Best effort" delivery of bundles from a source to a destination
  - Custody transfer + custody notification
  - Delivery notification, forwarding notification
- "Internal" services
  - Fragmentation of bundles (pro-active and re-active)
  - Bundle agent and bundle authentication + access control
  - Address compression (as URIs may get large)
- Security is another discussion
- Protocol: simple, binary protocol w/ efficient encoding of variable length fields
- Convergence layers: available for TCP, Bluetooth, LTP, …, files, …
- Running code available

DTN @ TKK Comnet (1)

- New course S-38.3151

Delay-tolerant Networking

- Period I in 2008/2009
- 3 ECTS
- 2 lectures per week
- Assignments
  - One theoretical assignment
  - One coding assignment using the DTN reference implementation (C/C++, Java, ruby)

DTN @ TKK Comnet (2)

- Postgraduate seminar on

Challenged Networks
(with a strong focus on Delay-tolerant Networking)

- Period III (Spring 2008)
- 3 ECTS
- Presentation + written summary paper (10 – 12 pages IEEE style)
- Preparation + opposition
- Probably block-style with one intro + assignments and 1 – 2 days of presentations

Ad: Looking for 1 – 2 people

- Special assignment / internship + subsequent master thesis
- Context 1: EC FP7 Project CHIANTI
  - Disconnection-tolerant transport and application protocols
  - Protocol analysis, implementation, measurements & simulation
- Context 2: Finnish Future Internet initiative (ICT-SHOK)
  - Energy and resource management (not just) in delay-tolerant networks
  - Enhancing and working with our Opportunistic Networking Environment (ONE) simulator
- Job demands
  - Self-organized, targeted style of working; openness for discussions with others
  - Interest in understanding and questioning protocol and system designs
  - Creativity and conceptual thinking
  - Capability of coding and the desire to get things up and running (publish code!)
  - Capability to write (documentation, publications) and present your ideas

