18-452/18-750
Wireless Networks and Applications
Lecture 18: TCP and Applications on Mobile Devices

Peter Steenkiste

Spring Semester 2024

http://www.cs.cmu.edu/~prs/wirelessF24/

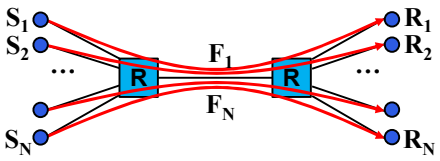
Impact of Wireless on Application Performance

- **Bandwidth sharing in the Internet**
- **TCP**
  - » **Basics**
  - » **TCP congestion control**
  - » **Impact of RTT**
  - » **Impact of Packet Loss**
  - » **Establishing a TCP session**
  - » **Maintaining TCP sessions while mobile**
  - » **Multi-path TCP**

Network Performance Properties

- Throughput: end-to-end bandwidth available between two communicating applications
  - » Depends on the bandwidth of the links on the network path
  - » How much of this bandwidth is available to the application, i.e., how is bandwidth shared on links
  - » Properties of the transport protocols used by applications using the links – how aggressive are they?
- Latency: time to send a packet end-to-end
  - » Depends on the propagation delay on all the links and the queueing delay on the routers
  - » Packet loss adds delay – impacted by transport protocols
  - » Queueing delay which is impacted by transport protocols - how aggressive are they?

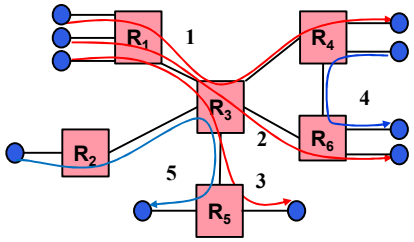
Bandwidth Sharing: Simple Scenario



- Dumbbell topology: N flows F_1-F_N share a bottleneck link with link capacity of B
  - » All access links have a more than enough capacity
- What would be a fair bandwidth allocation?

$$T(F_i) = B/N$$

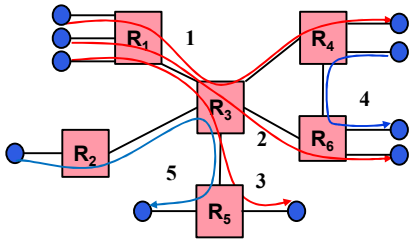
How About a More Interesting Network



- We have 5 flows that
  - » Use different paths in the network
  - » Use paths that have a different number of links
  - » Use links are shared with different numbers of flows
- Fair bandwidth allocation = equal bandwidth
- In this example, all flows get B/3

Is that reasonable?

Max-Min Fairness



1. Identify the link that is the most constrained
   - » Link R1-R3 supports 3 flows; flow 1, 2, and 3 get B/3
2. Subtract the assigned bandwidth from the link capacities
   - » R1-R3 has 0 left; R3-R5, R3-R4, R3-5 have 2B/3 left; other links: B
3. Repeat steps 1 and 3
   - » Link R3-R5 supports two flows: flow 5 gets 2B/3
4. Keep Repeating: Flow 4 also gets 2B/3

Main TCP Functions

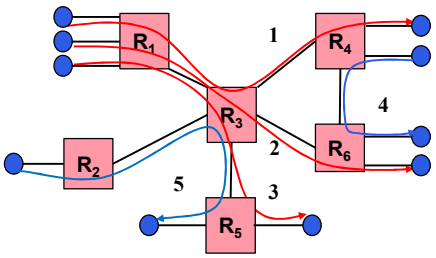
- Connection management
  - » Maintain state at endpoints to optimize protocol
  - » Introduces delay even if you only send 1 byte of data!
- Flow control: avoid that sender outruns the receiver
  - » Uses sliding window protocol – can limit throughput!
- Error control: detect and recover from errors
  - » Lost, corrupted, and out of order packets
- Congestion control: limits transmit rate to avoid that senders flood the network
  - » Lack of congestion control leads to inefficiency and possibly network collapse
  - » Very hard problem – was not part of original TCP spec!

Flow and Error Control

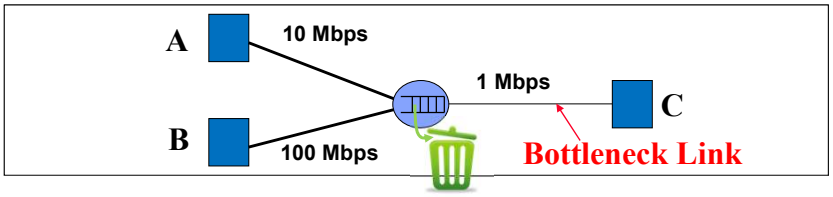
- Receivers may have limited space to store incoming packets
  - » Sliding window protocol avoids packet drops at receiver
- Receiver informs sender of how much buffer space it has available, limiting the transmit rate of the sender
  - » Throughput is limited to: window size/RTT
  - » Not a real concern on today's servers and end-points
- Lost packets must be retransmitted
  - » Retransmission is based on a time out, so delay can be significant
  - » May also delay packets that follow lost packets

Goals TCP Congestion Control

- The goal of TCP congestion control is to limit the transmit rates of senders so traffic can be handled efficiently by the network
  - » Similar to traffic control on the road – avoid gridlock
- Ideally traffic will get a fair bandwidth allocation
  - » Fair = equal bandwidth under the same conditions



TCP Congestion Control 101



- The bottleneck limits the throughput of senders A and B to receiver C
  - » It is congested: there is more traffic than bandwidth

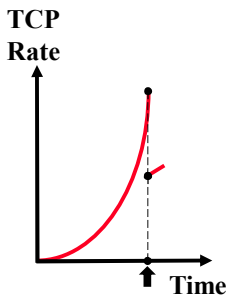
- What should the router do?
- It drops packets – what else can it do?
  - » Informally: when the queue is full, it overflows

- What should senders do?
- Slow down when there is congestion
  - » Congestion event = packet loss

Loss

- The Internet design and TCP specifically assume that packet loss is a sign of congestion
  - » It is defined as a "congestion event" and TCP will reduce its transmit rate
- This is appropriate in wired networks since practically all losses are the result of queue overflow
- However, wireless channels are more challenging which can result in higher packet loss rates
  - » This was a big problem in the early days of WiFi
- Solution: wireless network aggressively avoids packet loss on the wireless ink
  - » To higher level protocols, the wireless link looks like a wired one!
  - » WiFi and cellular use hybrid ARQ: forward error correction and retransmissions if needed to recover from errors

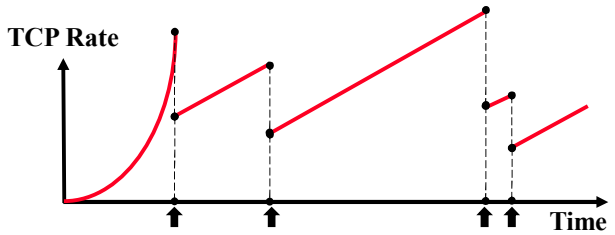
What Transmit Rate Should a new TCP Session Use?



- Analogy: suppose you want to know how long it takes to drive to a new destination?
  - » It depends on traffic!
- TCP discovers the available bandwidth by increasing transmit rate exponentially!
  - » Double the transmit rate every RTT
  - » Goal: identify good transmit rate quickly
- This is called "Slow Start"
- Slow Start ends when the sender observes a congestion event
  - » Typically packet loss

What Rate Should TCP Use after Slow Start?

- What goals and constraints should be considered?
  - The sender wants to go as fast as possible!
  - Senders must slows down in response to congestion
- Continuously probe for more bandwidth
  - Increase the transmit rate slowly: typically by one MTU per RTT
    - MTU = Maximum Transfer Units (max packet size)
    - RTT = Round Trip Time
- Reduce rate when there is a congestion event
  - Cut the transmit rate in half

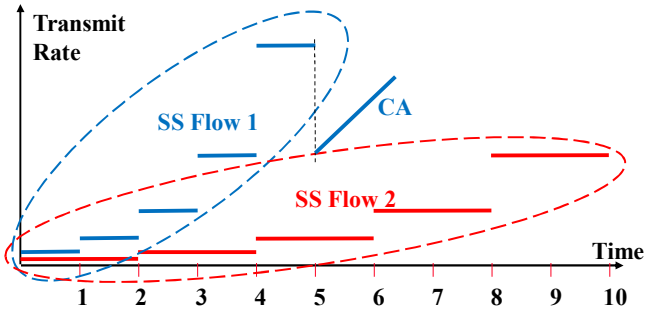


Relevance to Wireless Networks?

- The RTT of a TCP connection has a significant impact on throughput
- During Slow Start
  - All flows start with the same initial window, e.g., 10 MTU
  - But the rate increase depends on the RTT: a factor of 2 increases per RTT
  - With a low RTT, rate increases significantly faster!
- During Congestion Avoidance mode
  - The window size is increased by 1 MTU per RTT
  - Increase is faster when RTT is lower
  - Rate decrease is the same for all flows: cut in half

Example

- Flow 1 (blue) has an RTT that is half that of Flow 2 (red)
- SS: Flow 1's rate increases four times as fast as Flow 2
  - » It initial transmit rate is twice as high and it increases twice as fast
  - » Flow spends less time in SS, when rate is lower
- Congestion avoidance: rate increases twice as fast



Implications for Congestion Avoidance

$$\text{Rate} = \frac{MSS}{RTT} \times \frac{C}{\sqrt{P}}$$

MSS Maximum Segment Size
RTT = round trip time
C = constant depends on context
P = packet loss rate

- In Congestion Avoidance mode, the transmit rate is inverse proportional to the roundtrip time
- Again, flows with high RTT are at a disadvantage!
  - » Informal reason: low-RTT flows increase their rate faster, i.e., more aggressively
- Moving servers closer to clients has many advantages:
  - » Transmit rates increase much faster during Slow Start
  - » Higher throughputs during Congestion Avoidance
  - » Shorter network paths may reduce significant network bottleneck
- Builds a strong case for edge computing

The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm, Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, ACM Sigcomm, 1997

Limitations

- The version of TCP congestion control describes so far is just the basics
- Today, many congetion control algorithms are used with different properties
  - » All use slow start at the beginning of a session
  - » Many sometimes interpret RTT increases at congestion events
    - – Increased RTT means that queues are filling up, which may be a sign of congestion
    - – The goal is to avoid timeouts
  - » They increase/decrease rates differently
  - » Etc.

Wireless Challenges for TCP

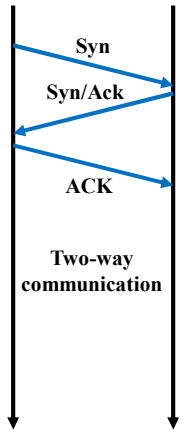
- Variability in available bandwidth, e.g., due to changes in channel CSI, handover, …
  - » Should be handled by congestion avoidance (later)
- Increases in latency due to MAC protocol and higher packet loss rate
  - » They have a surprisingly big impact on TCP!
- Loss of network connectivity, e.g., due to mobility
  - » TCP session breaks if the disconnection is long enough
  - » TCP state is lost, so new connection will have to be established when reconnected
  - » For application: not clear how much data was successfully transmitted

Impact of Wireless on TCP Congestion Control

- Variability in available bandwidth, e.g., due to changes in channel CSI, handover, …
  - » TCP congestion control needs to adapt more often
- Increases in latency due to MAC protocol
  - » WiFi: uplink and downlink transmissions use a shared channel with contention-based access
    - – Delays depend on how busy the channel is
    - – Exponential backoff can add significant delays
    - – Much less of a problem in switched ethernet
  - » Cellular: uses shared channels for uplink and downlink transmissions with schedules access
    - – Latencies have historically been high: 10s of msec
    - – Core network adds additional latency

Cost of Establishing a TCP Session

- TCP uses a three-way handshake to establish a TCP session
  - » Syn: provides parameters to needed for communication
  - » Syn/Ack: confirms Syn and establishes reverse path
  - » Ack confirms that the session has established
- TLS 1.2 adds another RTT
  - » Can be avoided based on earlier TCP session
- Adds significant latency to short data transfers
  - » In addition to the small congestion window during Slow Start



Mobility Breaks TCP Connections

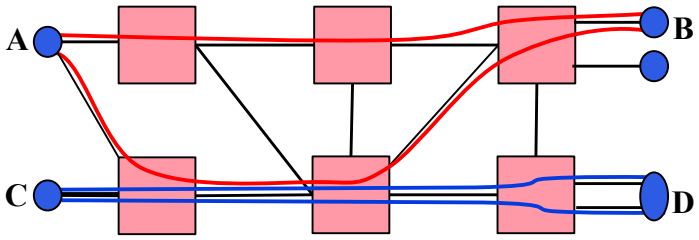
- Hosts use a 4 tuple to identify a TCP connection
  - » <Src Addr, Src port, Dst addr, Dst port>
- Changing either IP address of an endpoint breaks the TCP connection
  - » Host cannot determine which TCP session an incoming packet belongs to
  - » Will send packet to the wrong destination IP address
- The problem impacts both nomadic and mobile users
- TCP was not designed to handle this!

How to Make TCP Work with Mobility

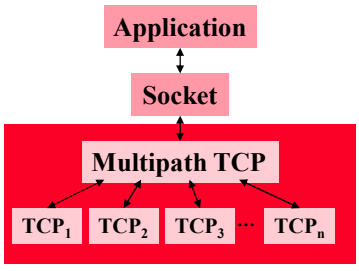
- You need to use a different way of associating incoming packets with TCP sessions
- General idea: add a level of indirection
- When the session is established, associate an "identifier" with the connection, e.g., session ID
  - » The session ID is included in packets – used to look up TCP state at the destination
  - » When a host moves to a new network and gets a new IP address, the TCP state is updated on both endpoints
  - » .. but the session ID remains the same
  - » This must be requires security
- Generally not supported for TCP, but
  - » Google's QUIC transport protocol does support mobility

Multiple-Path TCP
MPTCP

- Multi-path TCP allows a client to send packets to a destination over multiple paths
- The most common use case is when one or both end-points have more than one network interface
  - » The different network interfaces have different IP address
  - » They can represent a different source/destination endpoint



How Does MPTCP Work?



- The application uses one socket for the connection
  - » It uses traditional primitive for sending/receiving data
  - » There are extra commands for control
- The MPTCP connection combines regular TCP connections
- Error, flow, and congestion control is handled by the individual TCP connections
- A multipath layer implements the multi-path abstraction
  - » Implements adding and dropping paths
  - » Distributes traffic load over the single path session

Using MPTCP

- MPTCP has several interesting use cases
  - Networks that are unreliable and or unstable (e.g., bandwidth fluctuations)
  - Recovering from network failres
- Increasing overall throughput was not an explicit target
  - A lot of research on making MPTCP behave fairly with respect to TCP
  - Interesting research but not entirely clear whether this is a high priority for users
- MPTCP can help with maintaining connectivity for mobile users
  - Mobile phones today support both WiFi and cellular
  - Users uses cellular, but sign

MPTCP and Mobility

- MPTCP can help with maintain connectivity for mobile users
  - Mobile phones today support both WiFi and cellular!
- Example: user uses WiFi while mobile
  - Gets new IP address in new network and may be disconnected while switching
  - Solution:
    - before losing connectivity with WiFi 1, add a second TCP path over cellular
    - Cellular path supports TCP session after loss of connectivity to WiFi 1
    - Find new WiFi network, WiFi 2, and connect
    - Add a second path over WiFi 2 to MPTCP connection

Outline

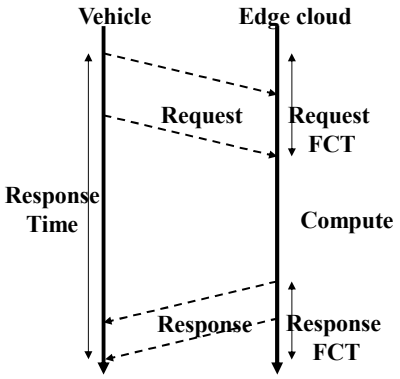
- Wireless and the Internet
- Mobility: Mobile IP
- TCP and wireless
- Applications and wireless

How Does Wireless Impact Applications

- The layered Internet protocol stack largely isolates applications from layer 1&2 details
- Except for:

1. Disconnected operation: it is impossible to hide that fact that the device is no longer connected to the network
   - » This is a big deal – not just a detail!
2. Variability in available bandwidth - TCP
   - » Due to changing channel conditions, handover, ..
3. Higher end-to-end latency (RTT) - TCP
   - » Due to the extra delay introduced by MAC mechanisms

Applications Care About Response Time



- "Flow Completion Time": time to deliver a data object
  - » Sometimes called an Application Data Unit (ADU)
  - » The unit of data relevant to users
  - » Image, data set, …
- The FCT can have a significant impact on the latency of distributed applications
  - » Example: Remote Procedure Call
- FCT depends on network bandwidth and latency
  - » And other factors

Flow Completion Time

$$FCT = S_A / T$$

- Where:
  - » S_A is the size of the ADU
  - » T is the throughput of the end-to-end TCP connection
- The throughput T of the network connection depends on many factors
  - » Available bandwidth and the end-to-end latency
  - » For short data transfers
    - – Most or all of the data set is sent in Slow Start mode
    - – Establishing the TCP session also adds delay
  - » See previous lecture for details

Optimizing Content Delivery

- Content delivery is an important application of the Internet
  - » Video playback, web browsing, …
  - » Used widely on both mobile and stationary devices
- Retrieval of the content is driven by the client
  - » Follows an RCP model so network bandwidth and latency have a bit impact
    - – Good news: ADUs are typically large
- Examples:
  - » Video streaming: ADU is a video segment
    - – Key metric: bit rate
  - » Web browsing: pages consist of many web objects, many of which can be sent over a single TCP connection
    - – Key metric: Page Load Time (PLT)

How Can We Reduce FCT?

- For large transfers, network latency and bandwidth are a big impact
- Solution: Content Delivery Networks
- Replicate the content closer to users
  - » Users can retrieve content from a nearby CDN instead of a remote centralized data center
- How do users "find" the closest replica that has the content?
  - » DNS redirect: when using the domain name (x.com) to retrieve an IP address, the DNS server use the client's location to select the best CDN
- This is a general solution
  - » Not specific to mobile users

Adapting Web Content to the Device

- Mobile devices have smaller screens than computers, laptops, big screen TVs, …
- It makes sense to simplify the content
  - » Smaller images, lower video resolution, …
  - » Simplify the web pages: fewer embedded objects, …
  - » Saves both bandwidth and device processing time
- Many organizations use different web servers for mobile devices
  - » The HTTP protocol provides information that can be used by the provider to select the right web server and to adapt the content
  - » Some of the optimization is also done on the client
- Alternative use proxies that customize content
  - » Reduces the load on the mobile devices

Optimizing Video Delivery

- Key performance metrics: maximize bit rates while avoiding video stalls
- Video streaming adapts to the available network bandwidth
- Key idea: estimate available bandwidth based on delivery time of previous segments
  - » Segment i has an FCT of T_i and size S_i
  - » The available bandwidth is S_i / T_i
  - » All video segments are stored with multiple bit rates
  - » Assuming a fixed playback time, we can choose the highest bit rate segment that can be delivered in time
- This is a general solution

Impact on Mobile Users

- Wireless devices can be used anywhere and are often used while the user is mobile
- Users expect near-ubiquitous coverage: no matter where the user is, the cellular network should be available
  - » This requires extensive testing: "Can you hear me now?"*
  - » WiFi users do not have the same expectation
- Some environments are challenging
  - » High speed trains moving at 300 km/h +
  - » It actually works! With some minor kickups

* Old Verizon commercial
+ https://doi.org/10.1145/3230543.3230556

Intermittent Connectivity

- When a device is disconnected, applications can no longer access the Internet
  - » When laptops were first introduced, applications would sometimes just crash, but …
- Applications that rely on long-lived connection are problematic by design
- Alternative: REST APIs may help in handling short disconnections for some applications

REST(full) APIs

- Application programming interface for client server interactions that is:
  - » Clean separation of client and server using well defined data formats and interfaces
  - » Statelessness: each client request is independent, allowing the server to complete processing
  - » REST APIs are viewed a good software engineering practice since they simplify building systems from components
- Simple example is web interface: HTTP
  - » HTTP is stateless and supports a well defined set of requests clients can to servers
  - » Not just for page retrieval! Much more general
- In our (limited) context: avoids the user of long lived sessions, failed requests can be retried

* REST: representational state transfer

Disconnected Operation

- When a device is disconnected, applications can no longer access the Internet
  - » When laptops were first introduced, applications would sometimes just crash, but …
- Users who are disconnected want to continue to use their device
  - » Update calendar, read/write e-mail, edit files, ...
  - » This is true both on laptop and mobile phones
    - – But some applications are specific to laptops
- Here are some examples:
  - » Modifying and using files in a shared file systems
  - » Applications that use structured data
  - » Proactive services

Shared File Systems Example: Coda

- The Coda file system supports disconnected operation
- Coda is based on the AFS shared file system and allows disconnected users to work offline
- How does Coda work?
  - » Before disconnecting, users must replicates (cache) files of interest on their local device
    - – The files are available as "normal" files, e.g., they have their usual file name
  - » While disconnected, users can read and modify the local file copies
  - » Multiple users can have offline copies of the same file
- Since there are multiple copies, we need a replication strategy
  - » I.e., how do we maintain consistency

Replication Strategies

- Coda uses an "optimistic" strategy
  - » All users with a copy of the file can modify their local copy
  - » This creates a consistency problem
- With a "conservative" strategy, only one copy of the file can be modified while other copies are read-only
  - » This strategy is not practical, e.g., users need to know a priori what files they may need to modify and they need to know who gets the "file lock"
- The optimistic strategy raises a consistency issue

Coda Consistency

- When a user reconnects to the shared file system, Coda writes modified files back to the shared file system
- If only one of the copies has been modified, it keeps the most recent version
  - » This is consistent with the Unix file system semantics
- If both copies of the file have been modified the user is notified and needs to manually merge the changes
  - » Coda did provide some tools for common cases
- Does this model sound familiar?

Other Examples

- Andrew File System (AFS)
  - » Developed at CMU, starting point for Coda!
  - » Client acquires a lock when it caches a file – ~prevents simultaneous changes to files
  - » Clients can retrieve and change locked files, but they are made aware of the conflict – also responsible for merge
- Web based file sharing: box, google docs, …
  - » Models are all over the map
  - » Some provide fine grain consistency (e.g., google docs)
  - » Other provide simply create multiple copies of the file, i.e., the service they provide is transferring files between users
- Older file systems such as NFS ignored the problem
  - » Were designed for wired networks

Applications Using Structured Data

- Many distributed applications and services use structured data, not unstructured files
  - » Examples: To Do lists, agenda, e-mail, …
- The shared data can be viewed as "objects"
  - » Typed data structures with a specific format are only read and writing by a limited number of functions
- Example: e-mail is based on e-mail servers
  - » E-mail is created, sent, received, and used by e-mail clients
  - » Once it has been created, an e-mail is immutable
- As a result, e-mails can be read, composed, and deleted offline
  - » The e-mail client keeps a log of the e-mail activity and replays it once it is connected to the server
  - » Multiple users can even share an account

Other Applications

- Similar solutions can be used for other applications, e.g., calendars and to-do lists
  - » Tasks are appointments and tasks
- Unfortunately, data types are typically not immutable
  - » E.g., reschedule an appointment or delay a task
- The solution is to record changes and replay them once a devices reconnects to the server
  - » If there are no conflicting changes, merging is easy
- If multiple users make conflicting changes, they require manual resolution
  - » E.g., one user moves an appointment while another user canceled it

Proactive Services

- Some servers are proactive: they initiate operations, instead of just responding to client requests
- Examples
  - » Pushing new e-mails, new tasks, .. proactively to clients
- Challenge: mobile clients may have different IP addresses as they move around
  - » How does the server "find" them?
- Whenever a client reconnects to the network, it contacts the server
  - » Server can then give them updates and (possibly) start pushing updates proactively, or client polls regularly
- The problem is actually more complicated
  - » Network Address Translation boxes (NATs) often require that the client polls (FYI only) – clients needs to poll server periodically

Resource Constrained Mobile Device

- Mobile phones and wearable devices are resource constrained
  - » Limited power and compute cycles
  - » Many reasons: cost, weight, size, …
- For compute-intensive applications, computational offloading is an attractive alternative
  - » Use the huge, elastic resource pool provide by the cloud
  - » Clouds cycles are also cheap and offer statistical multiplexing
- Challenges:
  - » Partitioning the application, shared state
  - » Load on the network, network latency, cost, …

Edge Computing

- Offloading computing to remote clouds can result in high FCTs
  - » Bandwidth of WAN connections can be low
  - » High end-to-end latency hurts TCP performance
- Solution: edge computing
  - » Build smaller clouds (cloudlets) near population centers
  - » They can support low latency applications efficiently
  - » Edge computing is also used for virtualized cellular RANs
- Example: computational offloading for autonomous driving to edge clouds
  - » Silly view: a Raspberry Pi at each basestation
  - » Use nearby edge clouds (e.g., a few tens of miles)

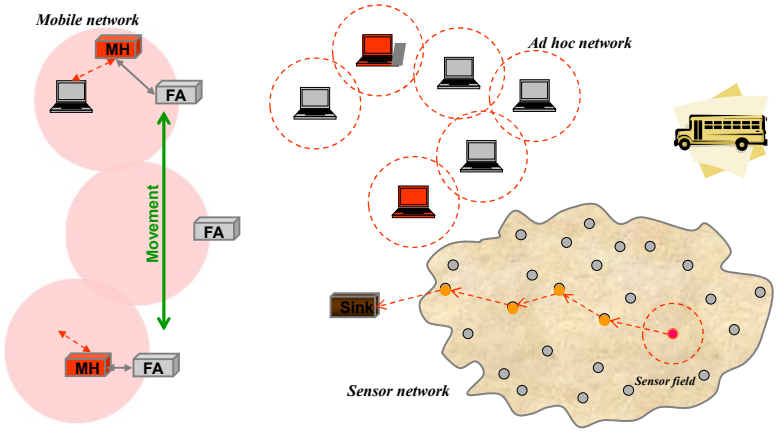
Outline

- Wireless and the Internet
- Mobility: Mobile IP
- TCP and wireless
- Applications and wireless
- Disruption tolerant networks

Challenged Networks

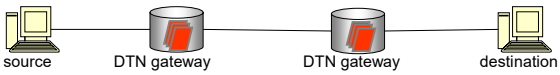
- Violate one or more of Internet's assumptions
  - » End-points may rarely/never be online at the same time
  - » Very long delay path, frequent disconnections, …
  - » Have naming semantics for their particular application domain
  - » Not be well served by the current end-to-end TCP/IP
- Examples
  - » Terrestrial mobile networks
  - » Some ad-hoc networks
  - » Sensor/actuator networks
- Goals for "disruption tolerant" networks
  - » Achieve interoperability between very diverse types networks
  - » Sometimes also called disruption tolerant

Background



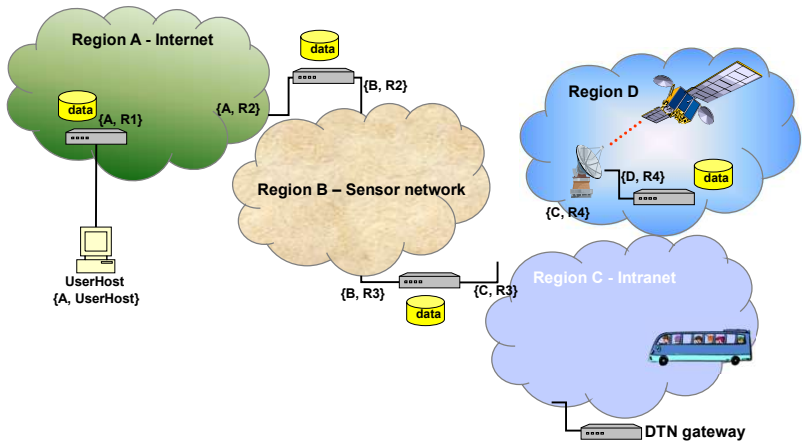
High-level Architecture

- Characteristics:
  - » Operate as an overlay above the existing transport layers
  - » Based on an abstraction of message switching
    - – Bundle
    - – Bundle forwarder (DTN gateway)
    - – Store-and-forward gateway function between different networks



- Constituent of DTN architecture
  - » Region: internally homogenous, i.e. same network stack, addressing, …
  - » DTN gateway: Interconnection point between region boundaries
  - » Name Tuple: {Region name, Entity name}

Example DTN



Finding Mobile Hosts: Two Simple Solutions

- Routing: mobile nodes keep "home" IP address and advertise route to mobile address as /32 in BGP
  - » Leverages LPM semantics - should work!!
  - » Bad idea: scalability
- DNS: mobile nodes get "local" IP address and update name-address binding in DNS
  - » DNS allows clients to update their address on the DNS servers of the address
- This should work but it is a terrible idea
  - » It results in a lot of write traffic to DNS
    - – Increases the load on the DNS servers
    - – Raises security concerns
  - » DNS relies heavily on caching of name-address pair
    - – Frequent updates reduce efficiency of caching

Old Slides