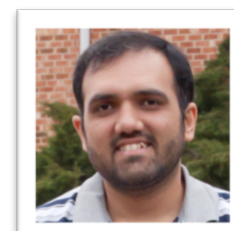
Cognizant Networks: A Model for Session-based Communications and Adaptive Networking

Jayabharat **Boddu**
Juniper Networks

Eric **Brown**
Office of IT, VT

Wuchun **Feng**
CS Dept., VT

Mark **Gardner**
Office of IT &
CS Dept., VT



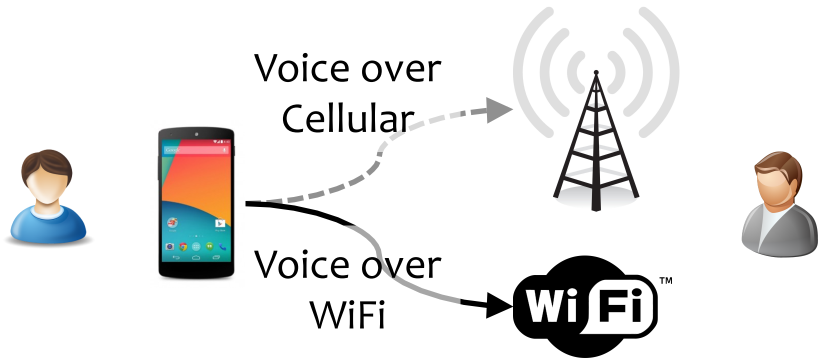
Umar **Kalim**
CS Dept., VT

VirginiaTech
Invent the Future

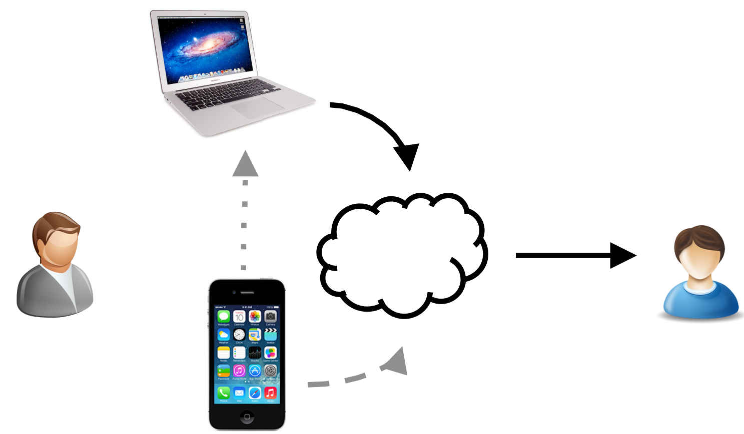
08/21/2015

SyNeRG
synergy.cs.vt.edu

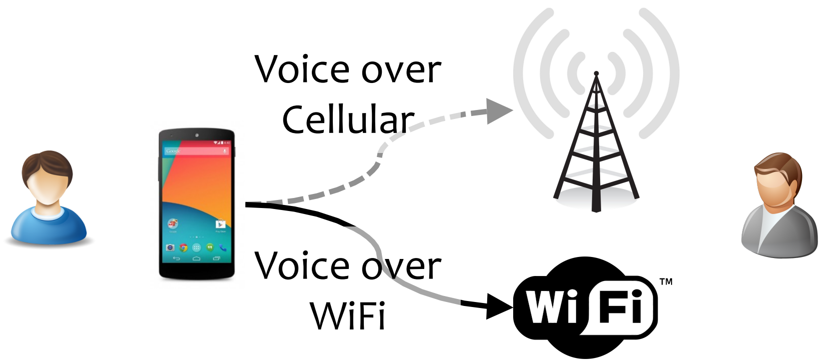
Modern Use Cases



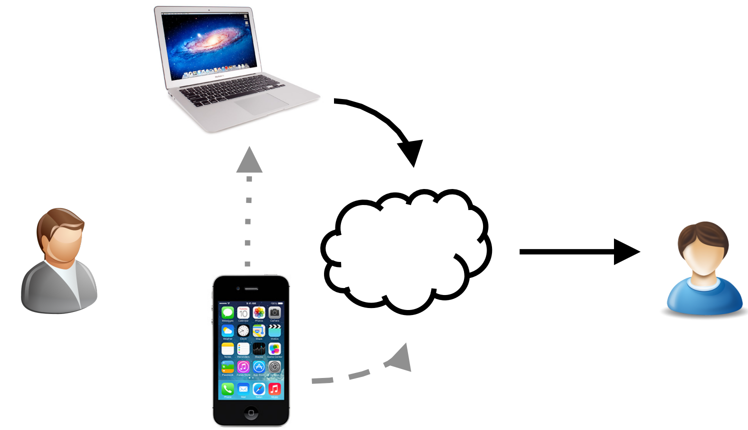
Seamless handoff w/ WiFi calls
(Android OS)



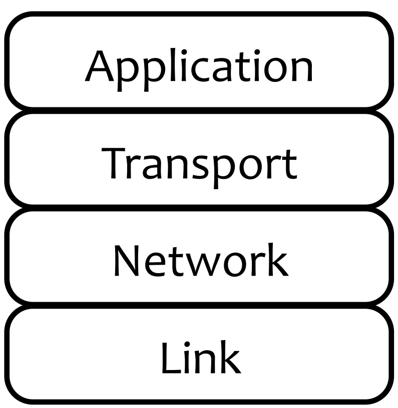
Continue using network
applications with different devices
(Apple Continuity)



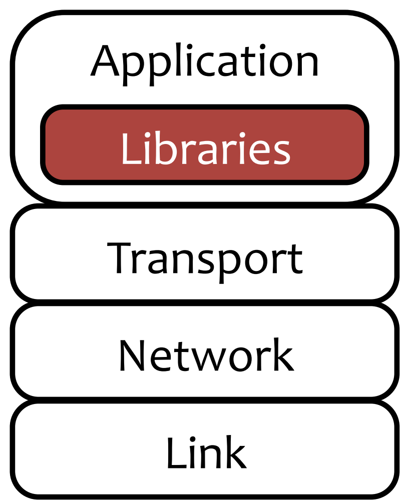
Seamless handoff w/ WiFi calls (Android OS)



Continue using network applications with different devices (Apple Continuity)

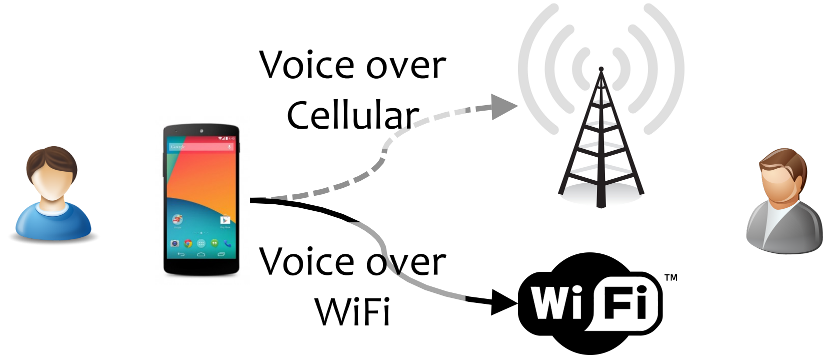


Typical TCP/IP Stack

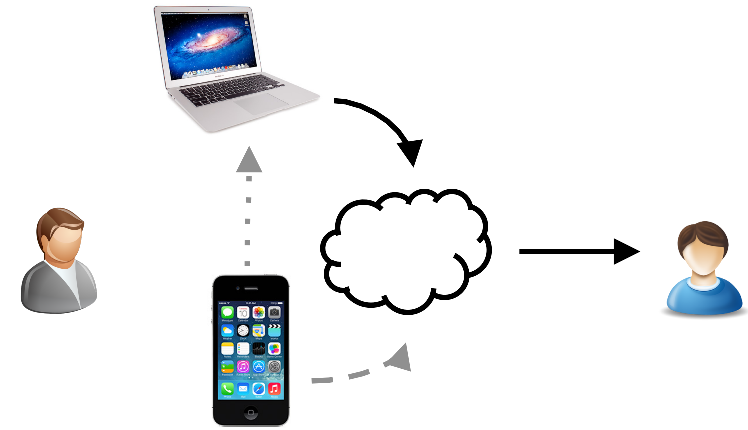


Modern Stack

Onus is on developers to implement <u>new features</u> as well as <u>support mechanisms</u>

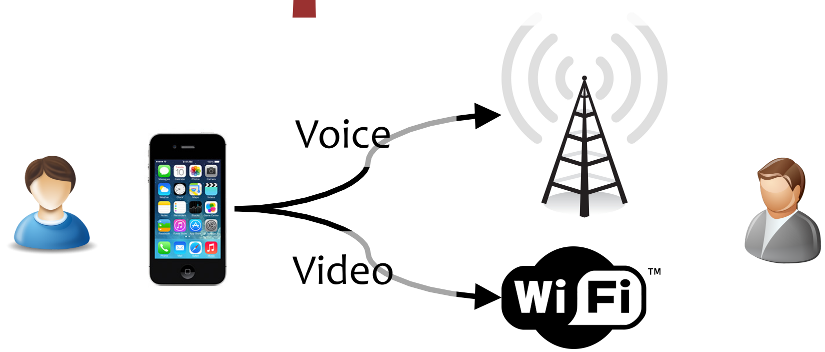


Seamless handoff w/ WiFi calls
(Android OS)

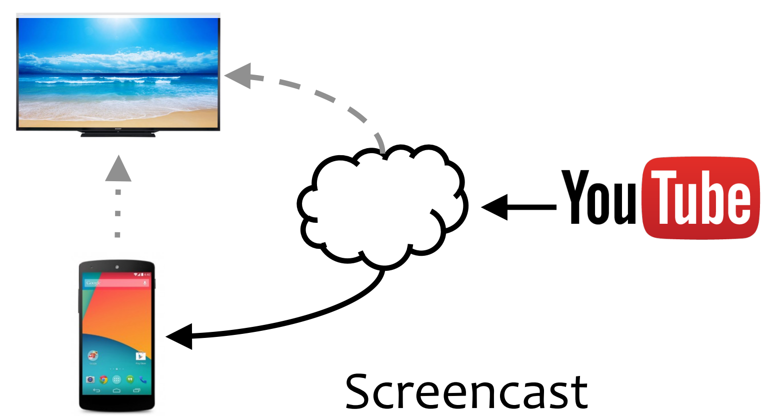


Continue using network
applications with different devices
(Apple Continuity)

Duplication of effort!



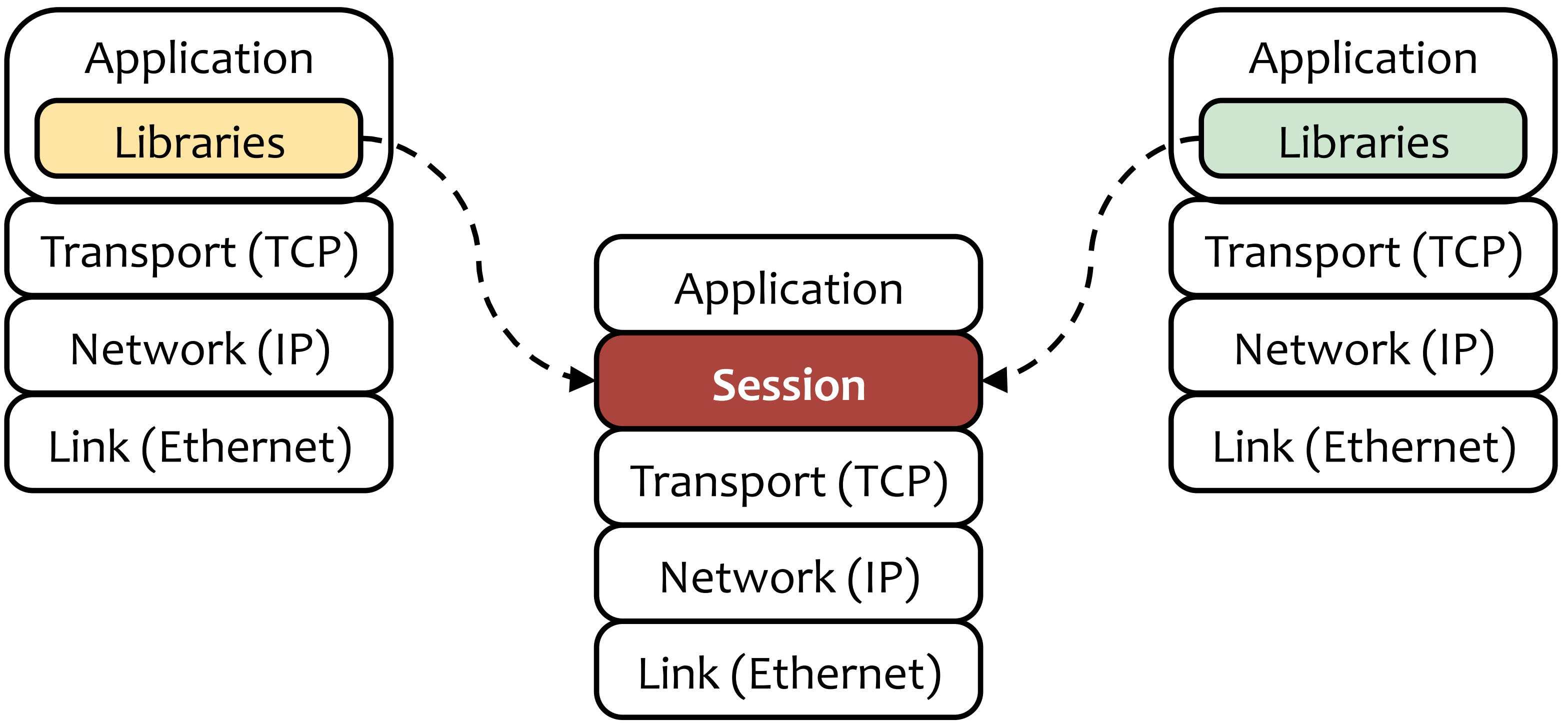
Voice & video over different networks
(Apple Facetime)



Screencast
(Google Chromecast)

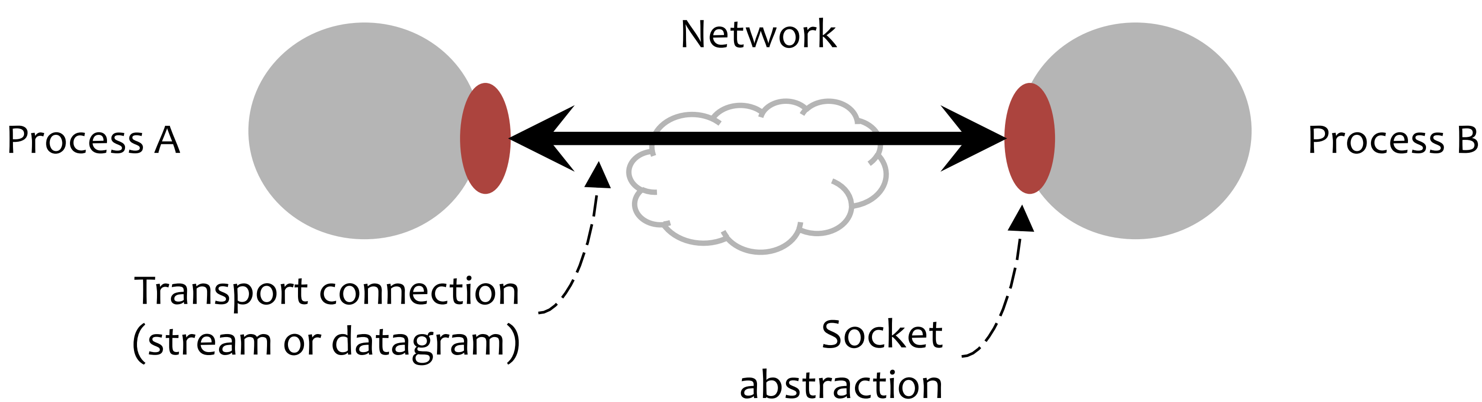
Aspiration

To enable support for modern use cases and future extensions



Challenges of Network Stack Extensibility

- Limitations of TCP (w.r.t modern use cases)
  - Session and transport semantics are coalesced
  - Model assumes two participants
  - Socket abstraction is limiting for modern use
    - Defines an endpoint of communication for a process
    - Label: <srcIP, dest IP, srcPort, destPort>



Limiting Assumptions of TCP Implementations

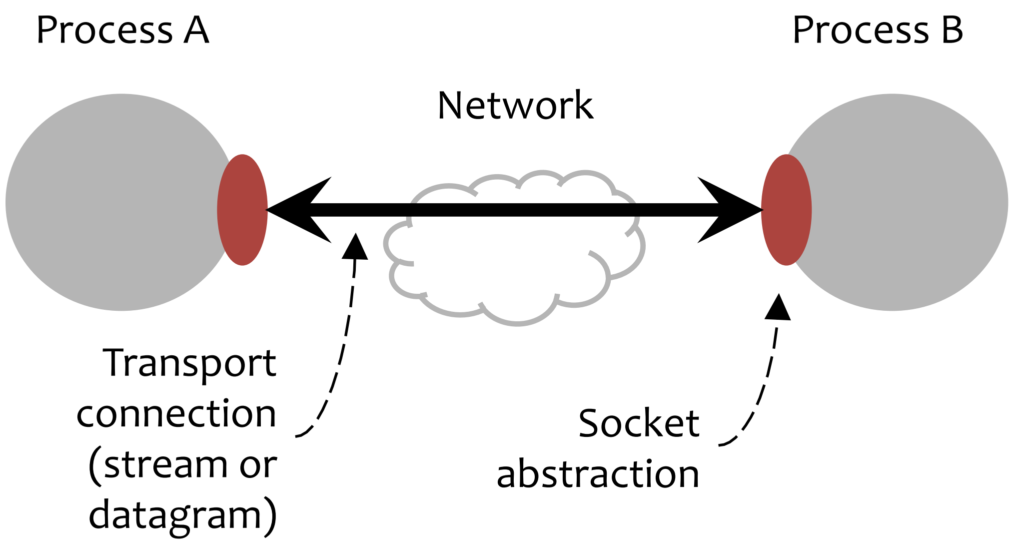
TCP Implementation Assumptions	Challenges
Stream label: socket pair Coupling of transport and network layer	• Naming

Single interface / attachment point
Preclude multiple transports via different networks
	• Flow Abstractions
	  • Multipath TCP
	  • Multihoming
	  • Flow Migration
	  • Hybrid Transport
	• Session Management

Lifetime of IP address assignment greater than lifetime of the connection	• Mobility
Dumb network	• Middleboxes as 1st class citizens
Static configuration	• Cross layer communication • Dynamic stack composition

Research Challenges

Communication Model



Socket defines an endpoint of communication
for a process identified by:
<srcIP, dest IP, srcPort, destPort>

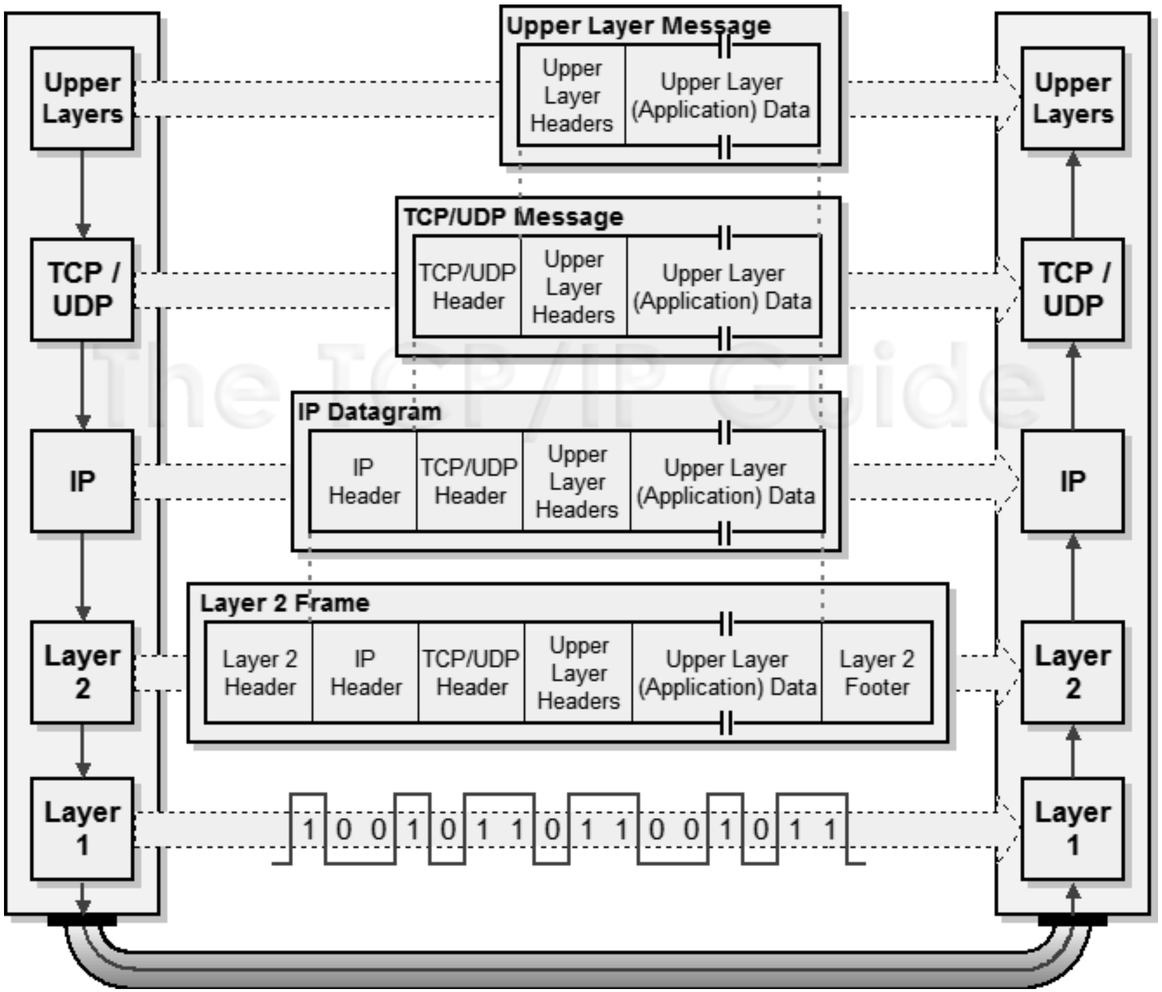
What abstractions do we need to *describe communication* and enable *management*?

How would these abstractions *fit* with existing designs?

How do we *discover*, *describe* and *use* context of communication?

How do we enable *dynamic configuration*?

Research Challenges
Backwards compatibility



How do we extend the network stack such that it allows *interaction with legacy stacks* and *applications*?

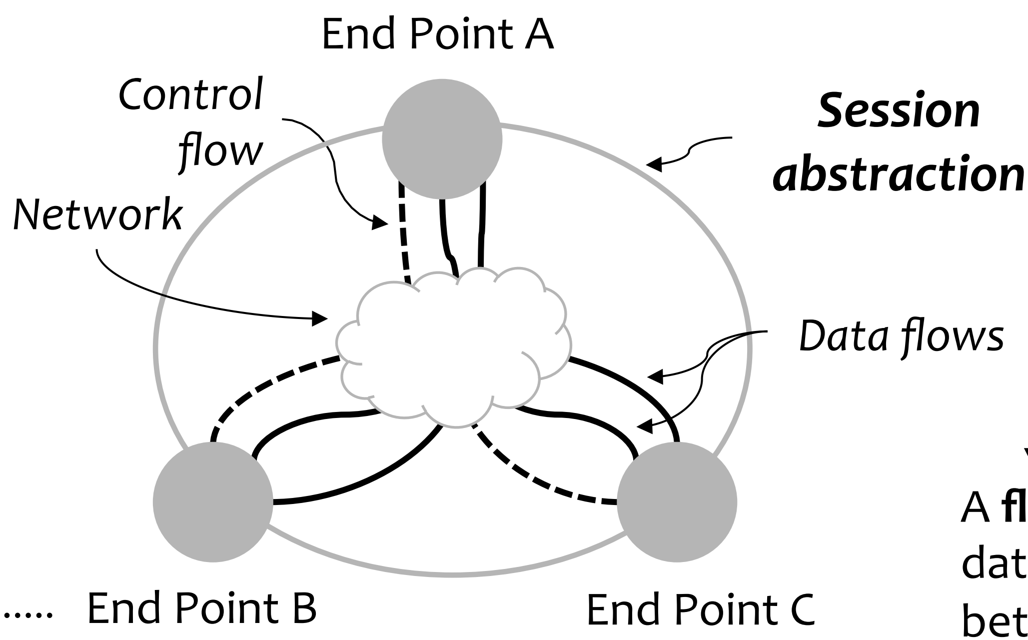
Image source: tcpipguide.com

Problem Statement

How to design, implement and evaluate abstractions that enable extensions to the network stack and allow configuration and reconfiguration of communications?

Session Abstraction

Representation of a conversation between participants in an agreed upon context

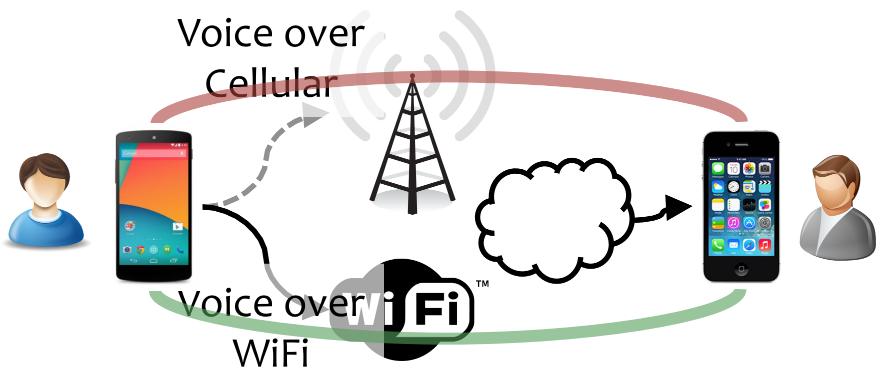


An **end point** is a participant that represents the source and destination of communications

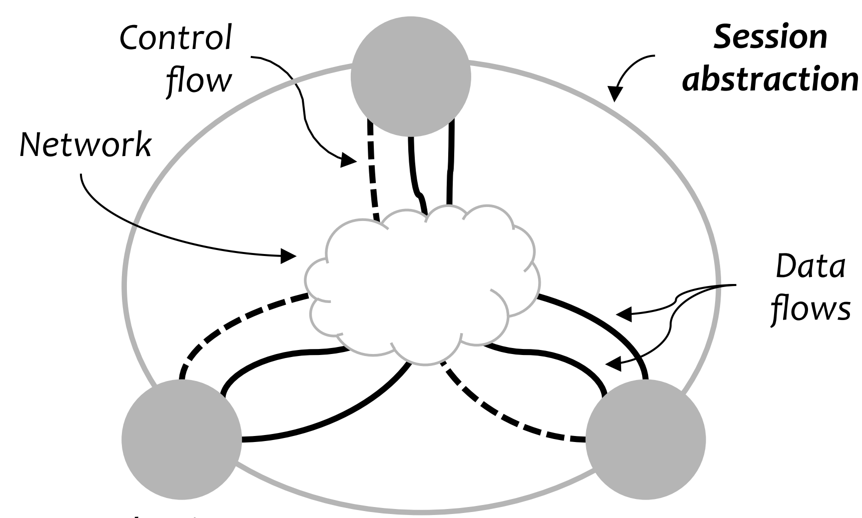
A **flow** represents a data exchange between a set of end points

Session Abstraction

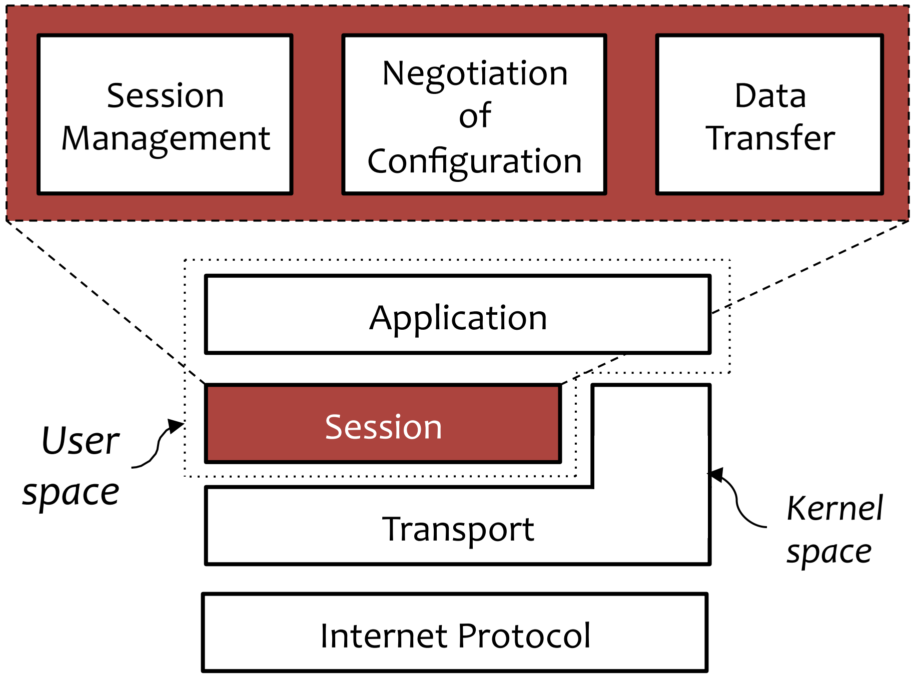
Representation of a conversation between participants in an agreed upon context



Seamless handoff w/ WiFi calls



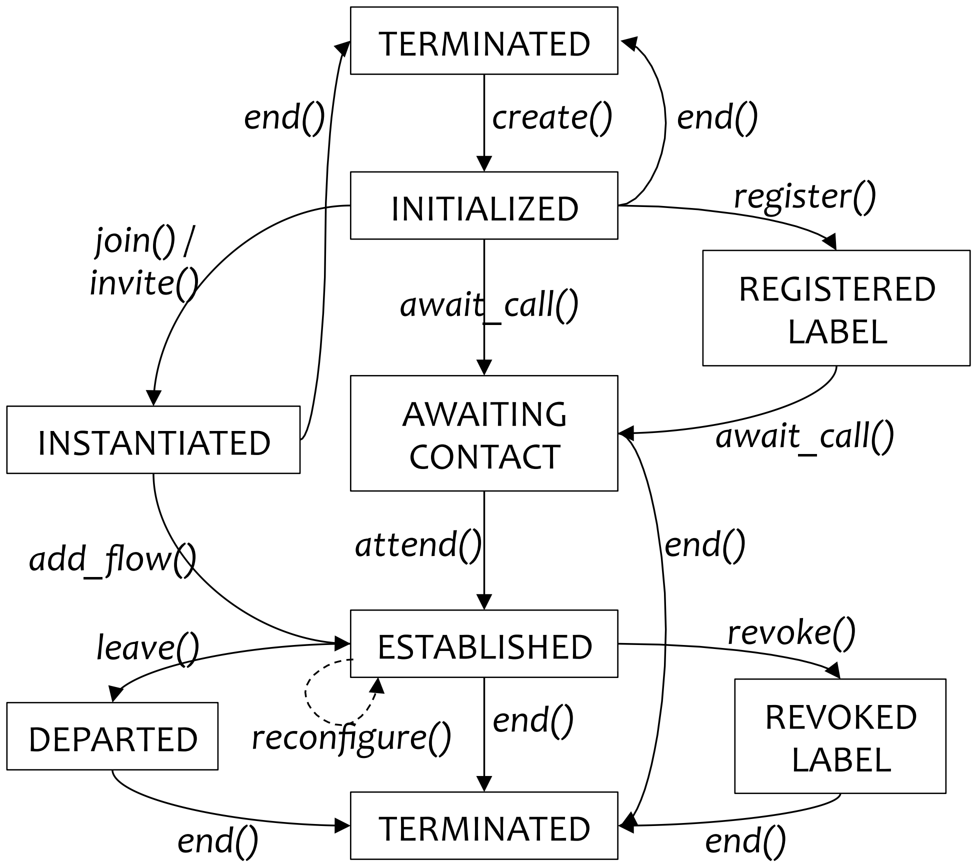
Session Layer in the Network Stack



- Additions to TCP to enable extensibility
  - Ability to accept custom TCP options to be aware of session-layer services
- Session-layer services in user space

Primitives and State Transitions

Session Management | Negotiation of Configuration | Data Transfer



Primitives and State Transitions

Session Management | Negotiation of Configuration | Data Transfer

Session-Related Primitives

create(session)
end(session)

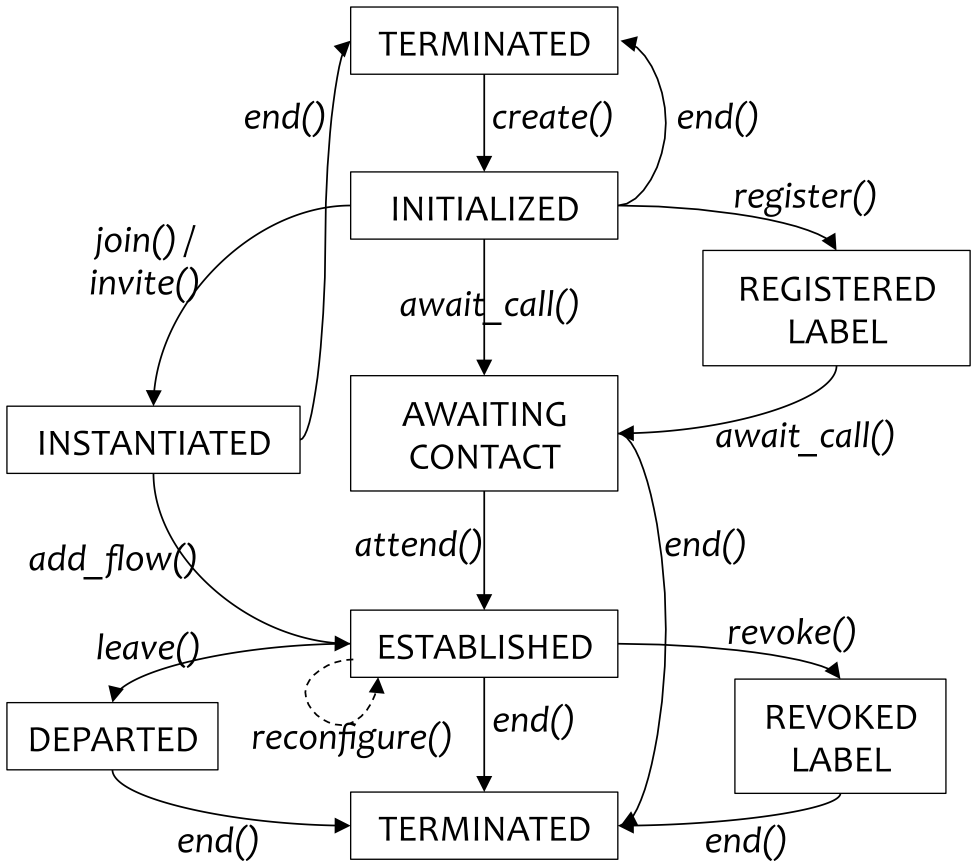
join(session)
invite(session, endpoint)
leave(session)

await_call(session)
attend(session)

reconfigure(session, verb)

register(label, endpoint)
translate(session, label)
revoke(session, label)

migrate(?)
suspend(?)



Flow-Related Primitives

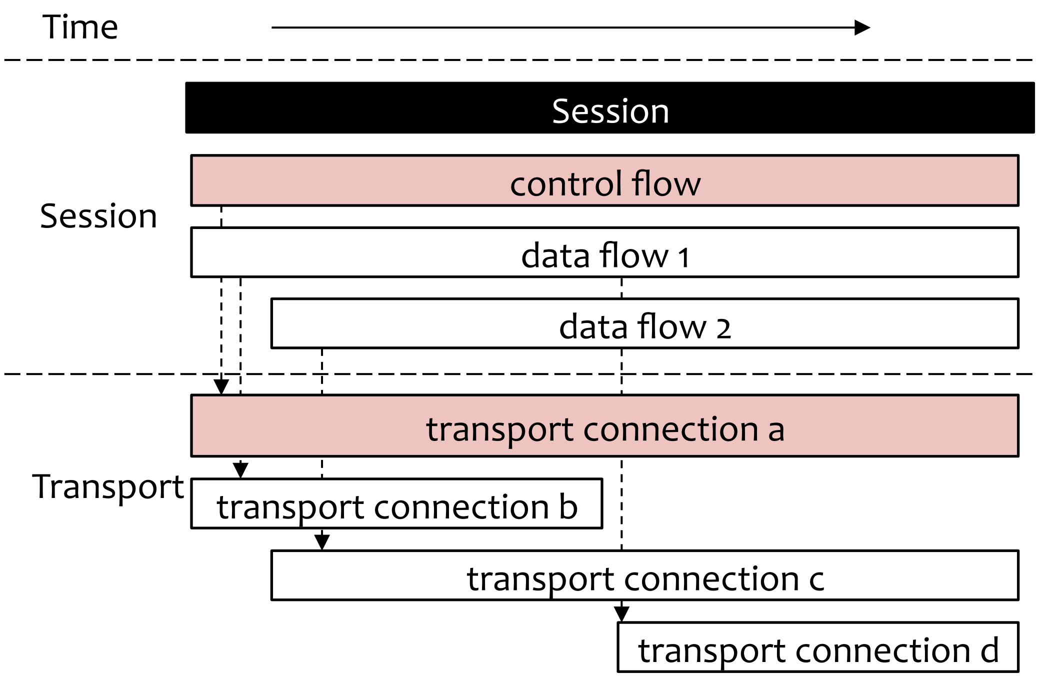
add_flow(session,
[structure,]
[type])
terminate_flow(session)

reconfigure(flow, verb)

read(flow)
write(flow)

Control Channel

- Enable exchange of control messages: verbs
  - Example: **refresh, suspend_flow, resume_flow**



Control Verbs

Session Management | Negotiation of Configuration | Data Transfer

```
{
    "VERB" : "refresh",
    "SOURCE" : "alice.node",
    "TRANSACTION ID" : "1872",
    "SESSION LABEL" : "a.session",
    "AUTH TOKEN" : "h@kl#S",
    "TIMEOUT" : 20
    "PAYLOAD": { "list":
        [{
        "END POINT LABEL" : "alice.node",
        "IP" : "192.0.1.222",
        "PORT": 5432
        }]}
}
```

What do Sessions contribute?

- Allows us to describe communications
- Allows management of conversation
  - Manage one or more flows and endpoints
- Enables negotiation of configuration
  - Example: Encrypted flows

- Avoids duplication of effort
  - Application of primitives on one or group of flows
    - Example: Apple, phone call over WiFi and cellular net.
  - Migration of session
    - Example: Chrome's Screencast

Summary & Plans

A Communications Model and Framework for Session-based Adaptive Networking

Model: Session-based Communication

1. Abstractions involving two or more participants [CONEXT'15]
   - Design: primitives and interactions
2. Separation of Session and Transport Semantics
   - Implications for communication patterns
3. Case Study: Context Awareness and Advantages [GLOBECOM'15]
   - Cascaded TCP [SC'12]

Realization: Enabling Extensions

1. Incremental Evolution [PFLDNeT'10]
   - How to extend and overcome limitations?
2. Reduce Duplication of Effort
3. Mobility & Virtual Machine Migration [ICCCN'13]
4. Resilient Transport [ICCCN'11]

Use: Dynamic Configuration Enabling New Paradigms

1. Control Protocol
2. Session-oriented communication paradigm
   - Middleboxes as first-class citizens (e.g., Engage firewalls, Inetd, tcp-wrapper)
3. Session based communication semantics
   - Privacy semantics per session (e.g., SSL)

Appendix

Related Work

	SCTP (2000)	TESLA (2003)	SST (2007)	SERVAL (2012)	MPTCP (2013)	Our ambition
Like TCP on the wire	✗	✓	✗	✗	✓	✓
Backwards compatible w/ apps	✓	✗ / ✓	✗ / ✓	✗	✓	✓
Backwards compatible w/ TCP stacks	✗	✗	✗	✗	✓	✓
Transport independent	✗	✗	✓	✗	✗	✓
Fault tolerant	✗	✓	✓	✓	✓	✓
Multihoming	✓	✗	✗	✓	✓	✓
Flow migration	✓	✗	✗	✗ / ✓	✓	✓
End point migration	✗	✗	✗	✗	✗	✓
Session abstraction	✗	✓	✗	✗	✗	✓

Related Work

	SCTP (2000)	TESLA (2003)	SST (2007)	SERVAL (2012)	MPTCP (2013)	Our ambition
Like TCP on the wire	✗	✓	✗	✗	✓	✓
Backwards compatible w/ apps	✓	✗ / ✓	✗ / ✓	✗	✓	✓
Backwards compatible w/ TCP stacks	✗	✗	✗	✗	✓	✓
Transport independent	✗	✗	✓	✗	✗	✓
Fault tolerant	✗	✓	✓	✓	✓	✓
Multihoming	✓	✗	✗	✓	✓	✓
Flow migration	✓	✗	✗	✗ / ✓	✓	✓
End point migration	✗	✗	✗	✗	✗	✓
Session abstraction	✗	✓	✗	✗	✗	✓

Perhaps ahead of its time

Related Work

	SCTP (2000)	TESLA (2003)	SST (2007)	SERVAL (2012)	MPTCP (2013)	Our ambition
Like TCP on the wire	✗	✓	✗	✗	✓	✓
Backwards compatible w/ apps	✓	✗ / ✓	✗ / ✓	✗	✓	✓
Backwards compatible w/ TCP stacks	✗	✗	✗	✗	✓	✓
Transport independent	✗	✗	✓	✗	✗	✓
Fault tolerant	✗	✓	✓	✓	✓	✓
Multihoming	✓	✗	✗	✓	✓	✓
Flow migration	✓	✗	✗	✗ / ✓	✓	✓
End point migration	✗	✗	✗	✗	✗	✓
Session abstraction	✗	✓	✗	✗	✗	✓

Great idea; wasn't adopted – lack of backwards compatibility

Related Work

	SCTP (2000)	TESLA (2003)	SST (2007)	SERVAL (2012)	MPTCP (2013)	Our ambition
Like TCP on the wire	✗	✓	✗	✗	✓	✓
Backwards compatible w/ apps	✓	✗ / ✓	✗ / ✓	✗	✓	✓
Backwards compatible w/ TCP stacks	✗	✗	✗	✗	✓	✓
Transport independent	✗	✗	✓	✗	✗	✓
Fault tolerant	✗	✓	✓	✓	✓	✓
Multihoming	✓	✗	✗	✓	✓	✓
Flow migration	✓	✗	✗	✗ / ✓	✓	✓
End point migration	✗	✗	✗	✗	✗	✓
Session abstraction	✗	✓	✗	✗	✗	✓

Perhaps mobile computing begins to flourish

Related Work

	SCTP (2000)	TESLA (2003)	SST (2007)	SERVAL (2012)	MPTCP (2013)	Our ambition
Like TCP on the wire	✗	✓	✗	✗	✓	✓
Backwards compatible w/ apps	✓	✗ / ✓	✗ / ✓	✗	✓	✓
Backwards compatible w/ TCP stacks	✗	✗	✗	✗	✓	✓
Transport independent	✗	✗	✓	✗	✗	✓
Fault tolerant	✗	✓	✓	✓	✓	✓
Multihoming	✓	✗	✗	✓	✓	✓
Flow migration	✓	✗	✗	✗ / ✓	✓	✓
End point migration	✗	✗	✗	✗	✗	✓
Session abstraction	✗	✓	✗	✗	✗	✓

Multiple interfaces in mobile devices are put to use

Related Work

TCP is here to stay,
perhaps sockets too

	SCTP (2000)	TESLA (2003)	SST (2007)	SERVAL (2012)	MPTCP (2013)	Our ambition
Like TCP on the wire	✗	✓	✗	✗	✓	✓
Backwards compatible w/ apps	✓	✗ / ✓	✗ / ✓	✗	✓	✓
Backwards compatible w/ TCP stacks	✗	✗	✗	✗	✓	✓
Transport independent	✗	✗	✓	✗	✗	✓
Fault tolerant	✗	✓	✓	✓	✓	✓
Multihoming	✓	✗	✗	✓	✓	✓
Flow migration	✓	✗	✗	✗ / ✓	✓	✓
End point migration	✗	✗	✗	✗	✗	✓
Session abstraction	✗	✓	✗	✗	✗	✓

Related Work

Lessons:

1) Avoid point solutions
2) Backwards compatibility is important

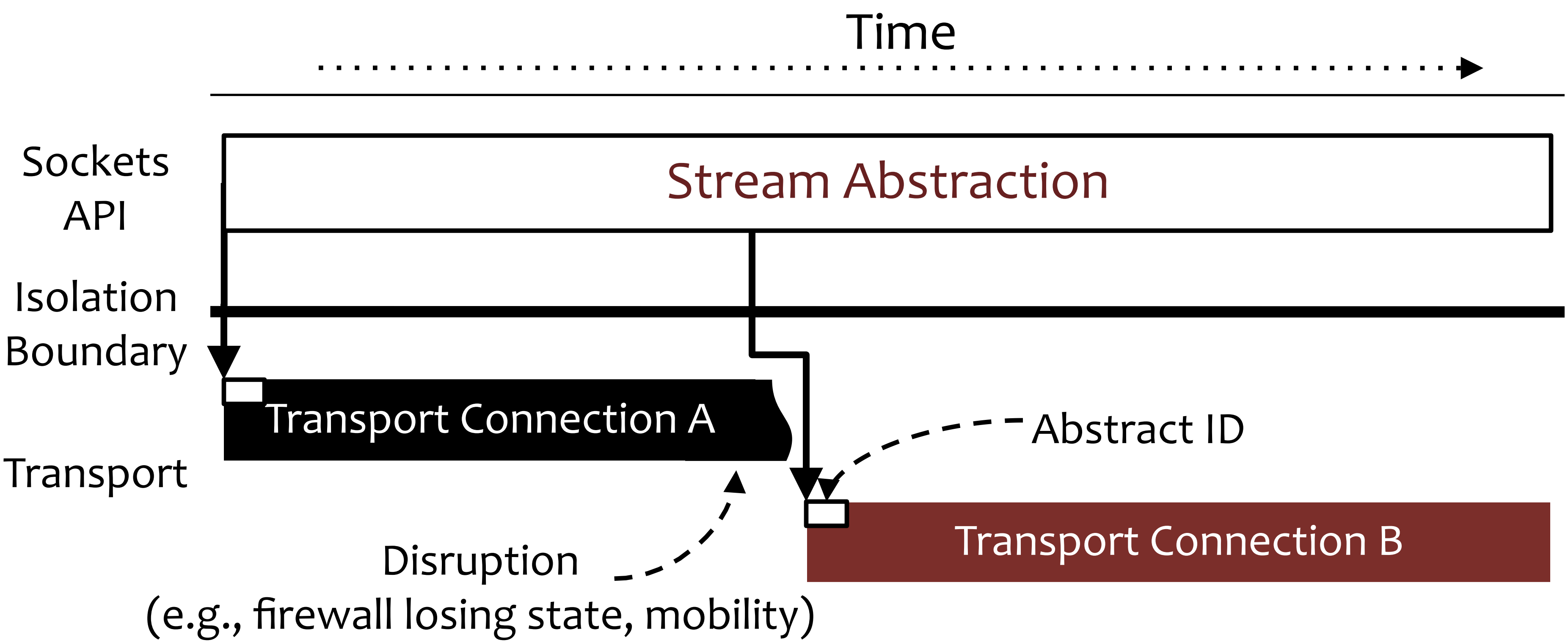
	SCTP (2000)	TESLA (2003)	SST (2007)	SERVAL (2012)	MPTCP (2013)	Our ambition
Like TCP on the wire	✗	✓	✗	✗	✓	✓
Backwards compatible w/ apps	✓	✗ / ✓	✗ / ✓	✗	✓	✓
Backwards compatible w/ TCP stacks	✗	✗	✗	✗	✓	✓
Transport independent	✗	✗	✓	✗	✗	✓
Fault tolerant	✗	✓	✓	✓	✓	✓
Multihoming	✓	✗	✗	✓	✓	✓
Flow migration	✓	✗	✗	✗ / ✓	✓	✓
End point migration	✗	✗	✗	✗	✗	✓
Session abstraction	✗	✓	✗	✗	✗	✓

Research Challenges

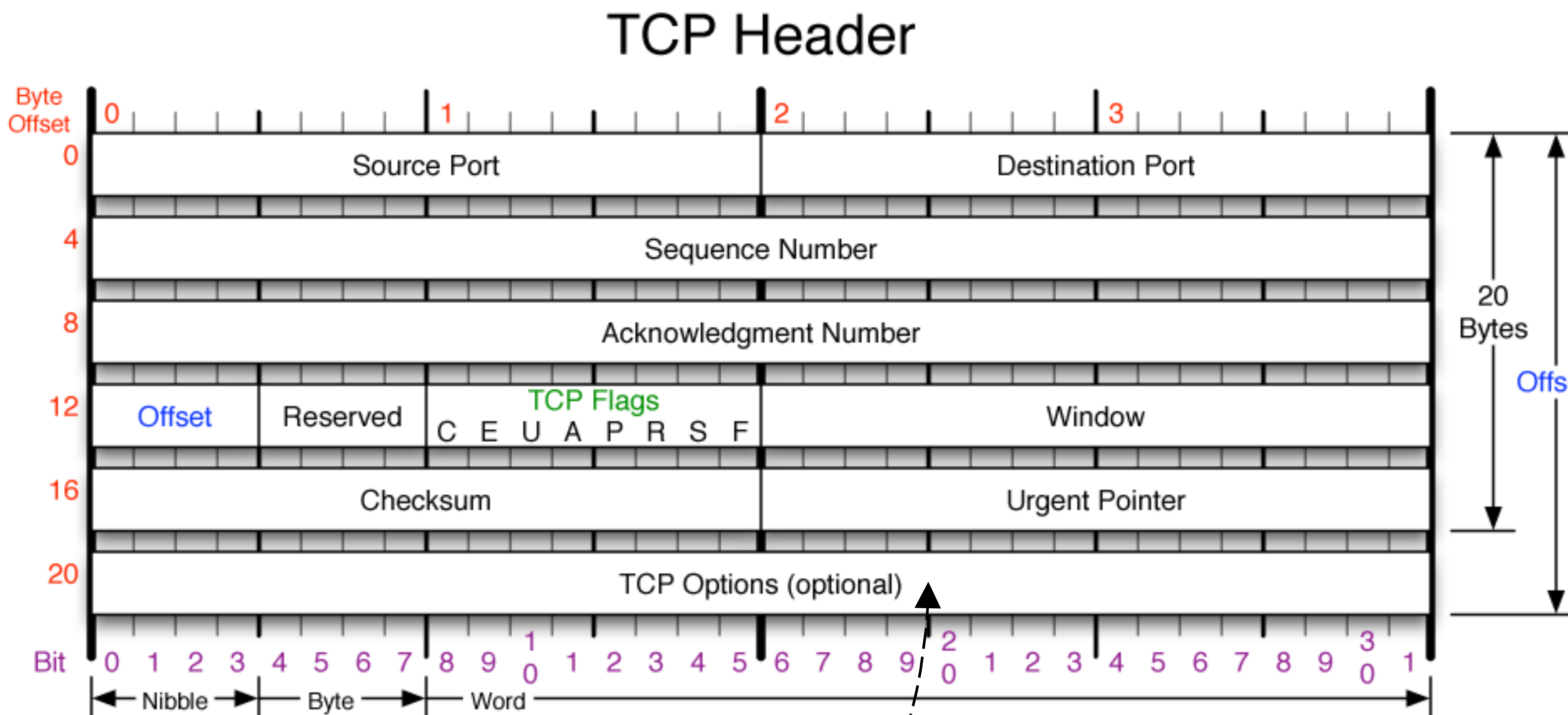
- Backwards-compatible extensions
  - How can we extend the existing network stack such that it allows *interaction with legacy stacks* and *applications*?
- Communication model
  - What abstractions do we need to enable *management of communication?*
  - How does *negotiation of configuration* enable context awareness?
  - How do these abstractions fit with existing designs?
- Framework
  - How do we develop a framework that enables *rich constructs*?
  - How do we *minimize duplication of effort*?

Logical Stream over TCP

- Application Stream flows between peers
  - Abstract ID: Independent of underlying transport



Backwards Compatible Extensions

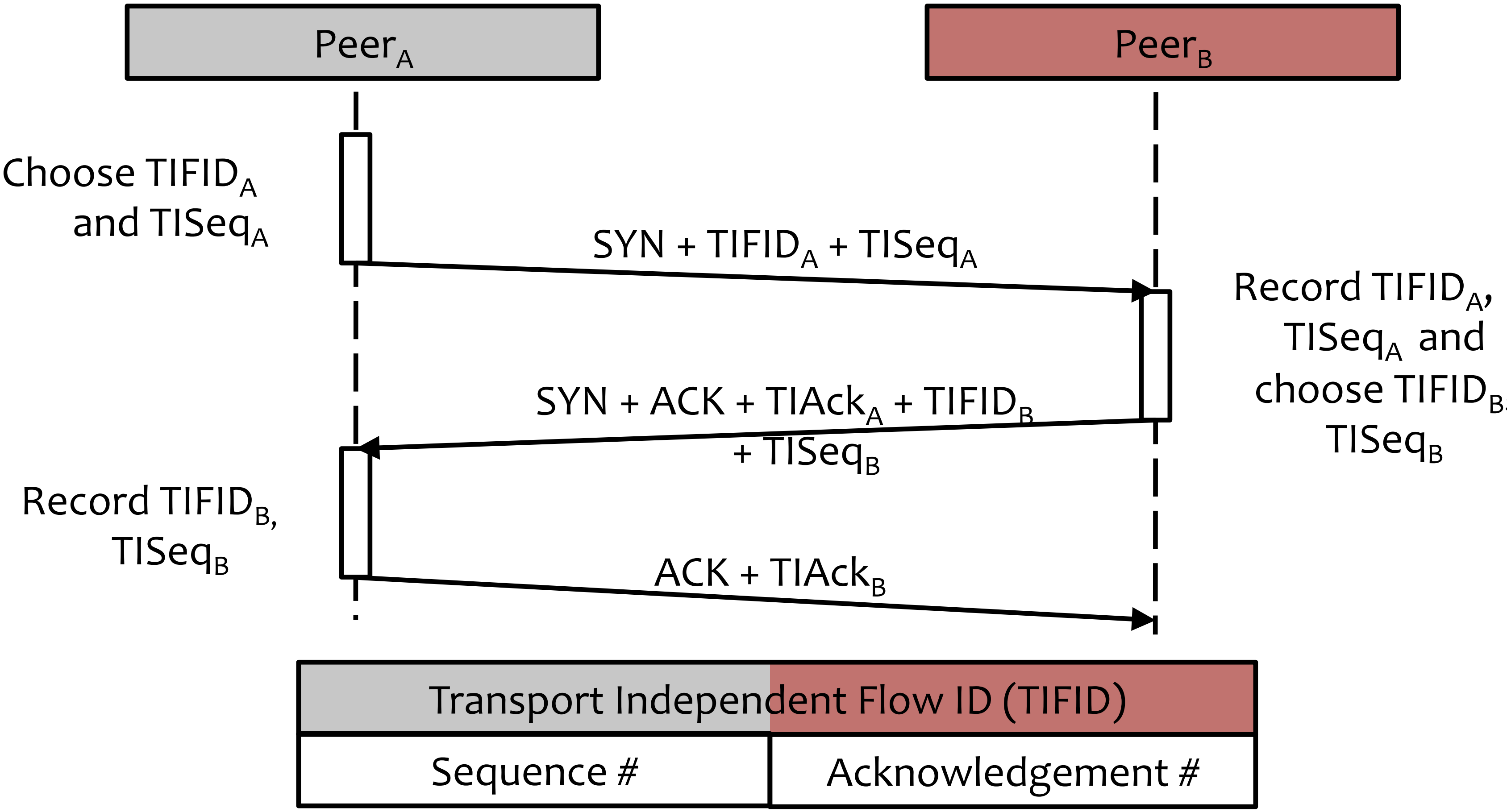


TCP Options space
(40 bytes)

20 bytes usable
(MSS 4, Window Scaling 3,
SACK 2, Timestamp 10)

TISeq – Transport Independent Sequence #
TIAck – Transport Independent Acknowledgement #

Backwards Compatible Extensions



TISeq – Transport Independent Sequence #
TIAck – Transport Independent Acknowledgement #

Definitions

- Communication
  - “a process by which **information** is **exchanged** between individuals through a **common system of symbols**, signs, or behavior.” (Merriam Webster Dictionary)

- Context
  - “The situation in which something happens; the **group of conditions** that exist where and **when something happens.**” (Merriam Webster Dictionary)

  - A **collection of attributes** that describe the **communication** and the **setting.** (our definition)

Why was state-of-the-art not adopted?

No killer application or disruptive technologies yet

Why was state-of-the-art not adopted?

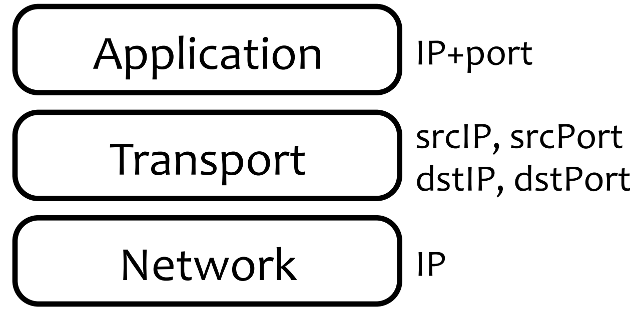
No killer application or disruptive technologies yet

What about:

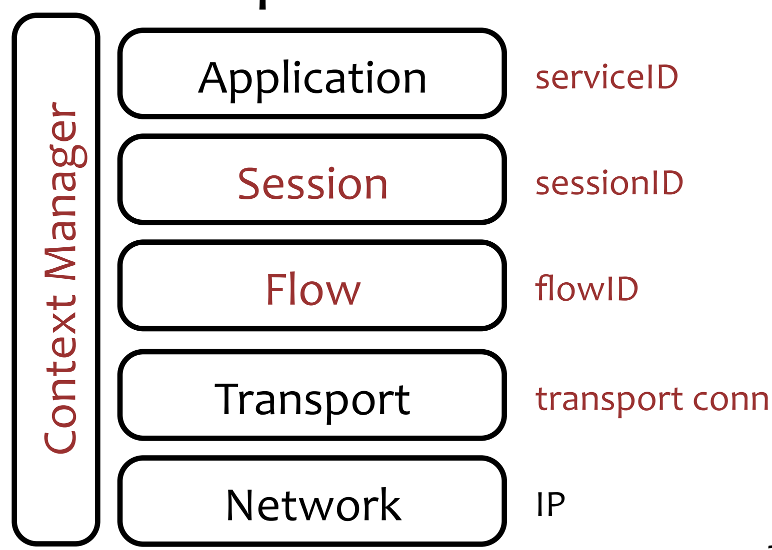
Abstractions

- **Data** plane abstractions
  - Clean separation of concerns
  - Richer set
    - Context
    - Event
    - Service
    - Session
    - Flow
    - Transport
    - Packet
    - Frame
    - (Network) Interface
    - Node
    - Facilitator
- Backwards compatibility

Overloaded Abstractions



Abstractions Redefined:
Clean separation of concerns

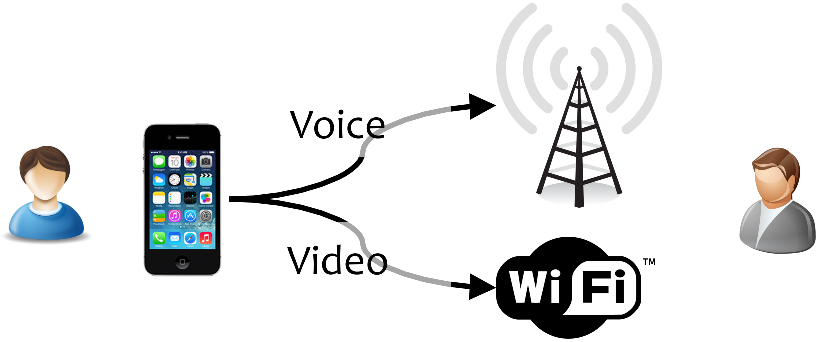


User Timeout Options Use Case

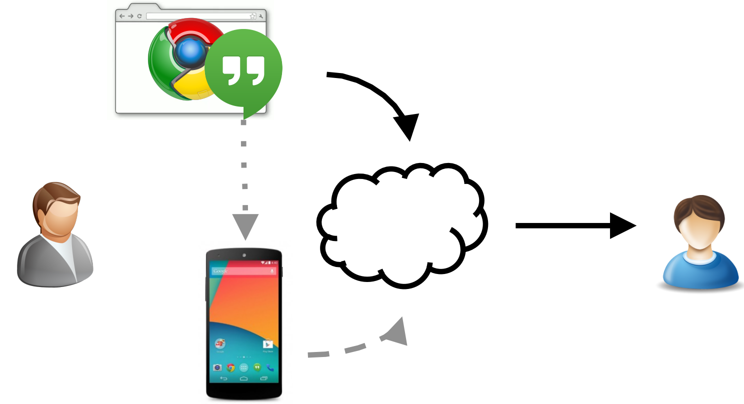
- Implementation
  - RFC 5482
    - TCP connection closed on timeout
      - No ACK before threshold
    - User Timeout Options
      - Define per connection timeout
      - Implemented using TCP socket options - SO_SNDTIMEO
    - Challenge for extension
      - Limited free space if typical TCP options are in use
        - Maximum Segment Size (4 octets)
        - Window Scaling (3 octets)
        - SACK (2 octets)
        - Timestamp (10 octets)
  - Publication discussing context manager
    - Abstractions, Architecture & API/interface and Implementation (UTO use case)

Long Term Goals

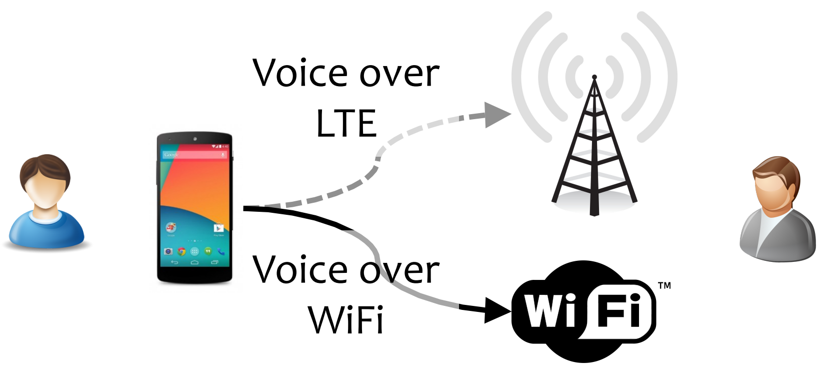
- A model that enables use of context in which communication happens
  - Abstractions & notion of communication context
    - Sessions, multipath comm., hybrid transport etc.
  - Architecture
    - Protocol design, functions, roles and interfaces
  - Framework
    - Prototype implementation and use-case applications



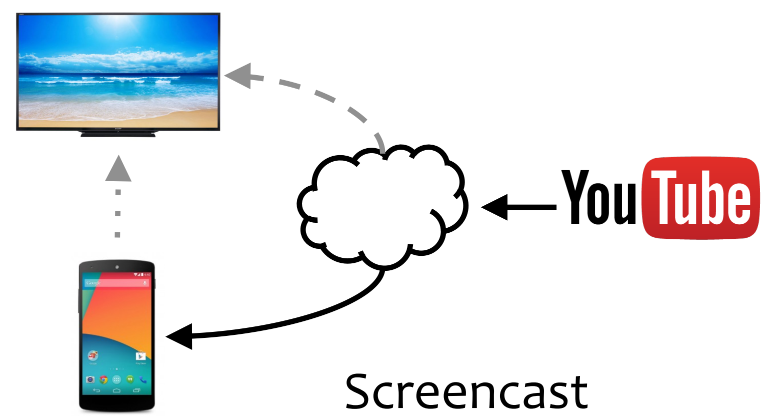
Voice & video over different networks with Apple Facetime



Call migration from browser to cell phone using Google Hangouts

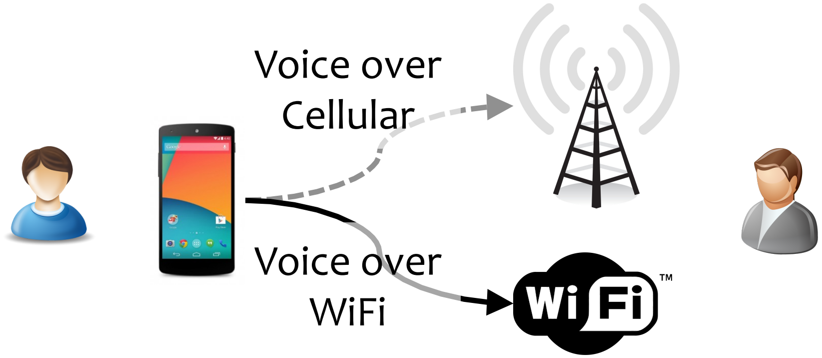


Seamless handoff w/ WiFi calls

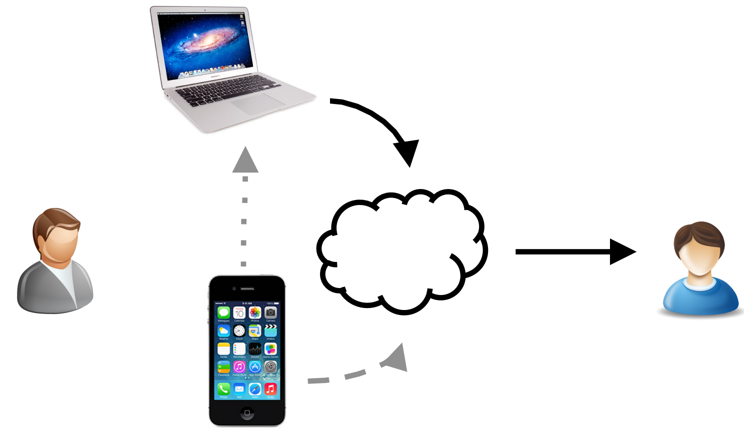


Screencast

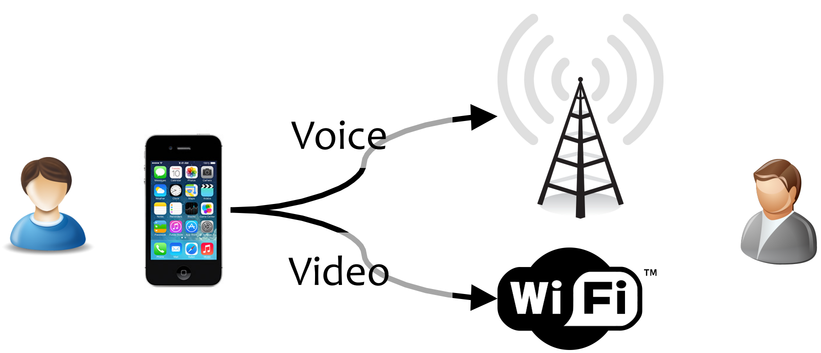
Icons downloaded from iconarchive.com



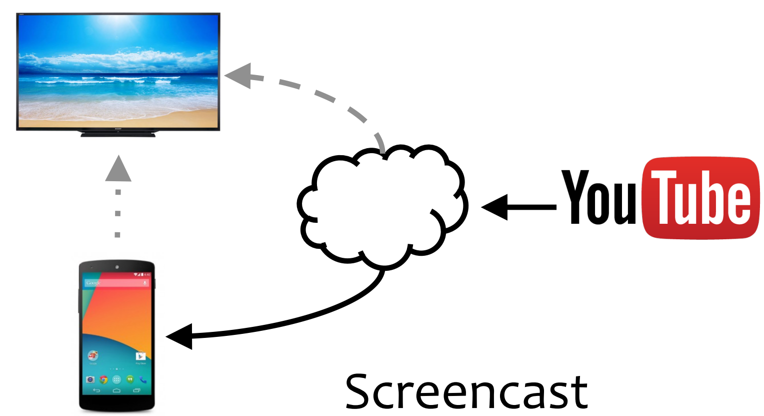
Seamless handoff w/ WiFi calls
(Android OS)



Continue using network applications with different devices
(Apple Continuity)



Voice & video over different networks
(Apple Facetime)



Screencast
(Google Chromecast)

Research Challenges

- Backwards-compatible extensions
  - How do we extend the network stack such that it allows *interaction with legacy stacks* and *applications*?

Research Challenges

Backwards compatibility?

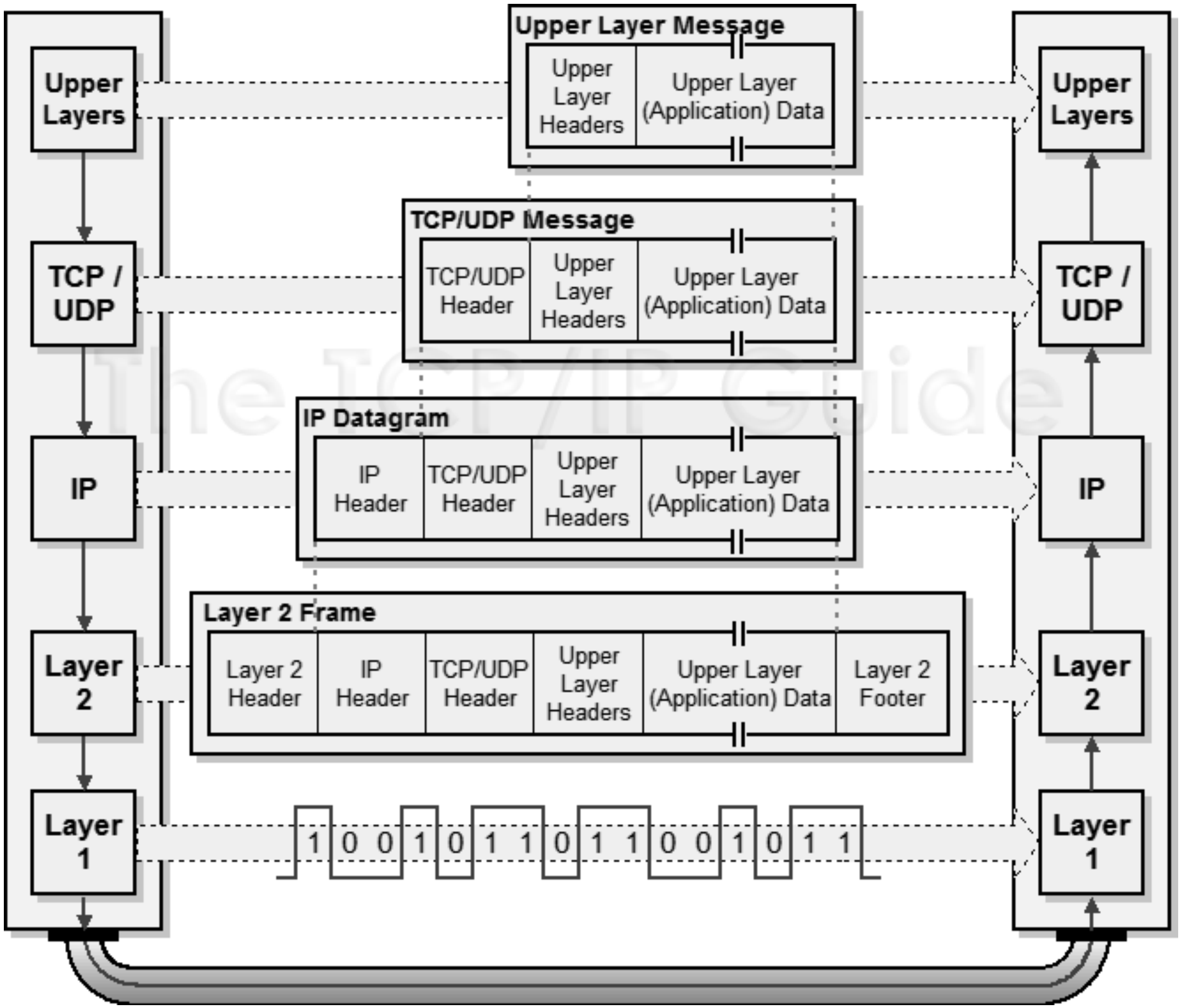
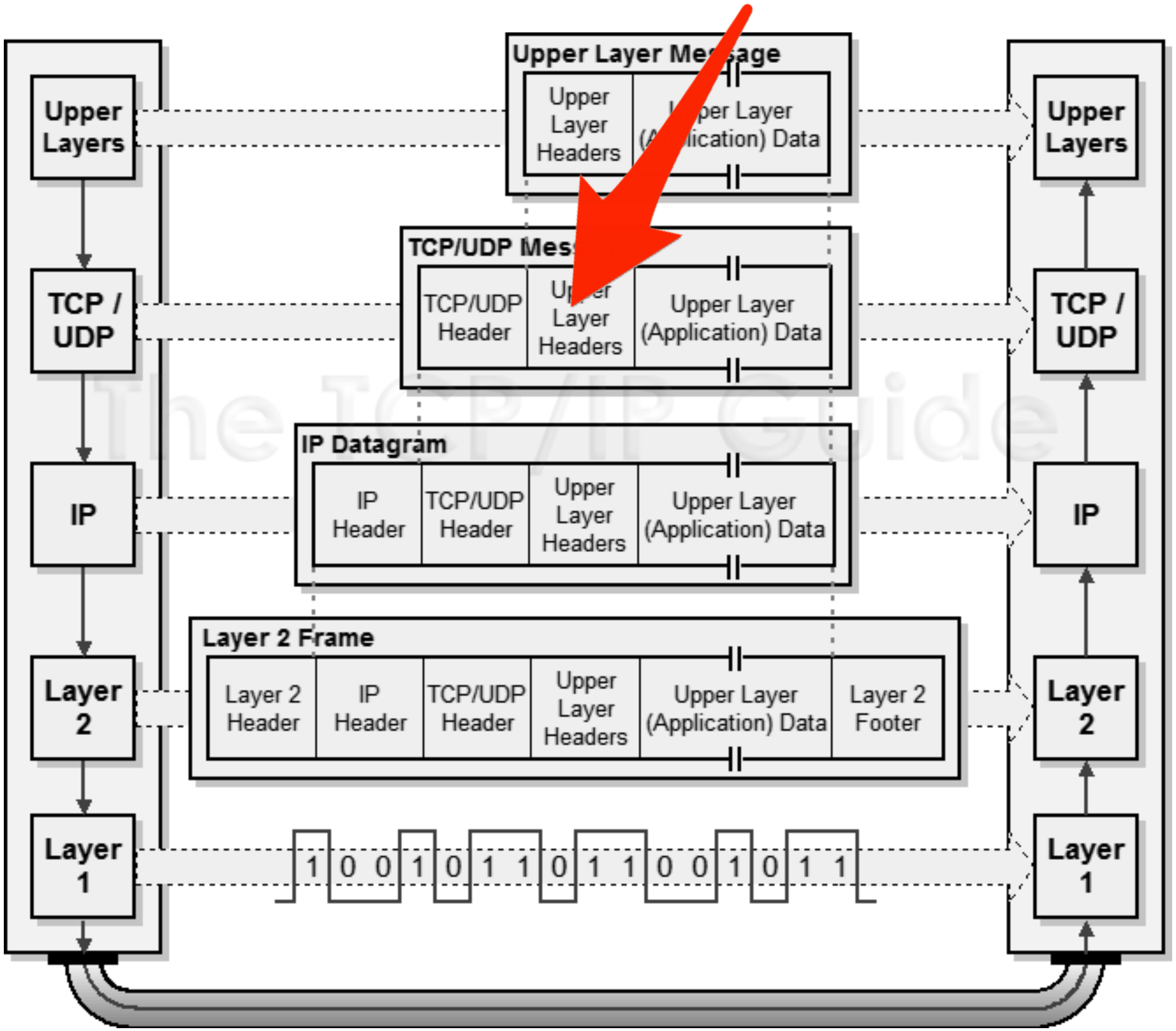


Image source: tcpipguide.com

Research Challenges

Backwards compatibility?

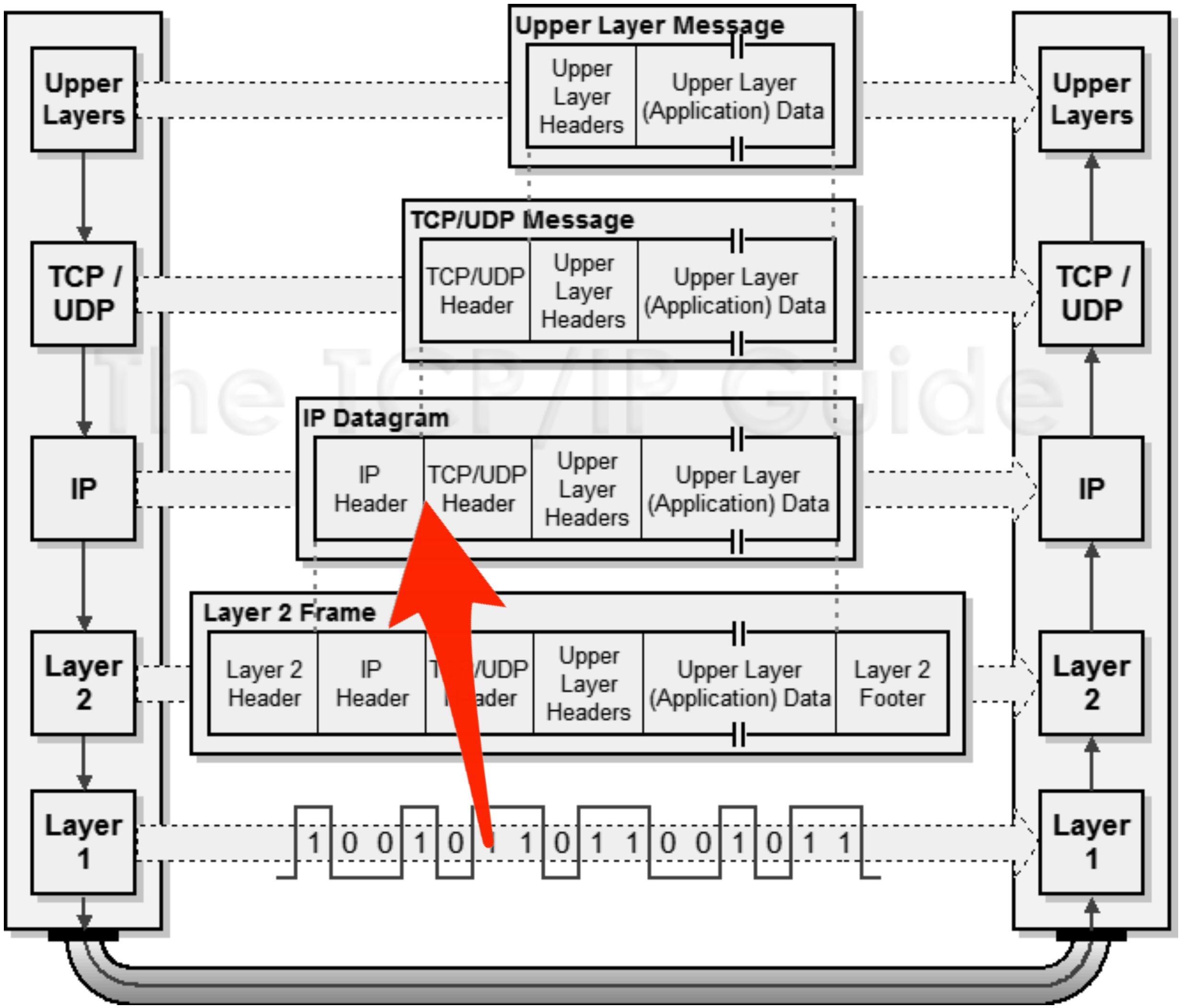


- Add layer above TCP?
  - *Legacy Stack will cause the app to fail!*

Image source: tcpipguide.com

Research Challenges

Backwards compatibility?

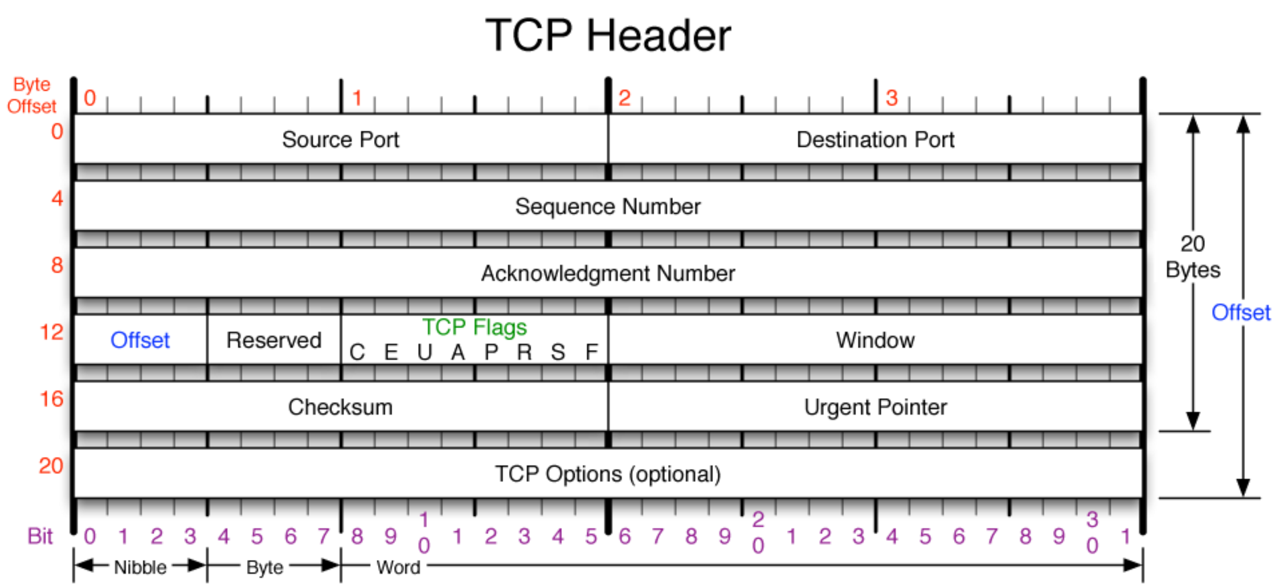


- Add layer above TCP?
  - *Legacy Stack will cause the app to fail!*
- Add layer below TCP?
  - *Middleboxes will not let the packets through!*

Image source: tcpipguide.com

Research Challenges

Backwards compatibility?



- Add layer above TCP?
  - *Legacy Stack will cause the app to fail!*
- Add layer below TCP?
  - *Middleboxes will not let the packets through!*
- Modify TCP?
  - *Legacy Stacks!*
  - *Middleboxes!*

Virginia Tech Invent the Future

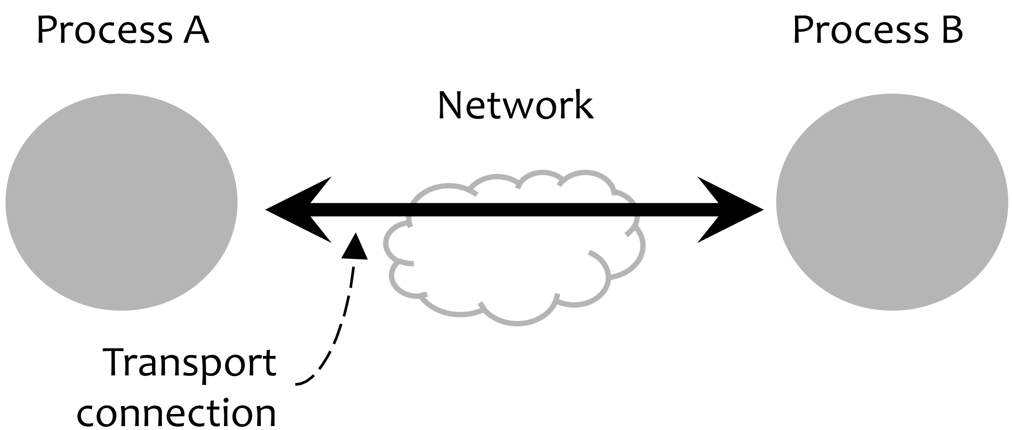
Image source: software-engineer-training.com

SyNeRG synergy.cs.vt.edu

Research Challenges

- Backwards-compatible extensions
  - How can we extend the existing network stack such that it allows *interaction with legacy stacks* and *applications*?
- Communication model
  - What abstractions do we need to enable *management of communication?*
  - How does *negotiation of configuration* enable context awareness?
  - How do these abstractions fit with existing designs?

Research Challenges Communication Model?

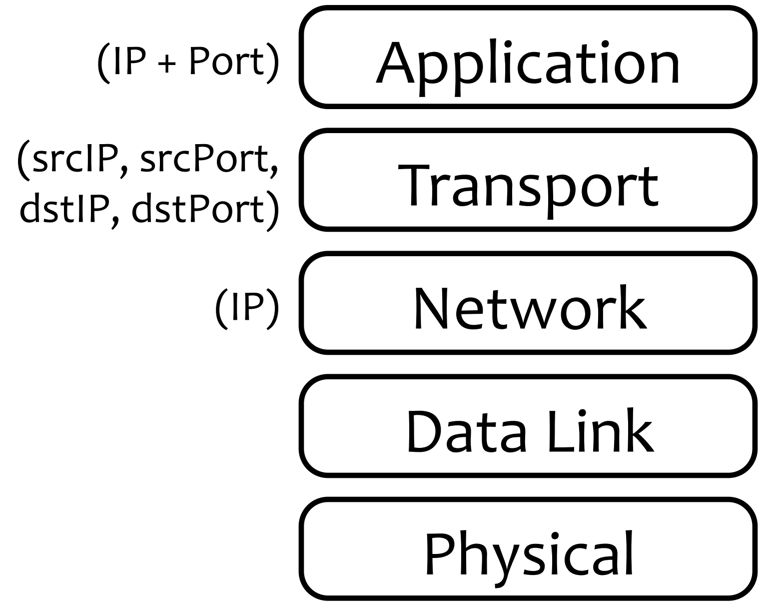


- Transport connection
  - Stream
  - Datagram

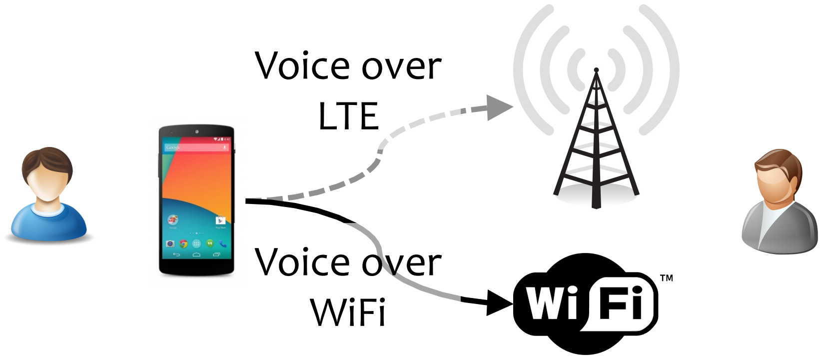
Research Challenges

Communication Model?

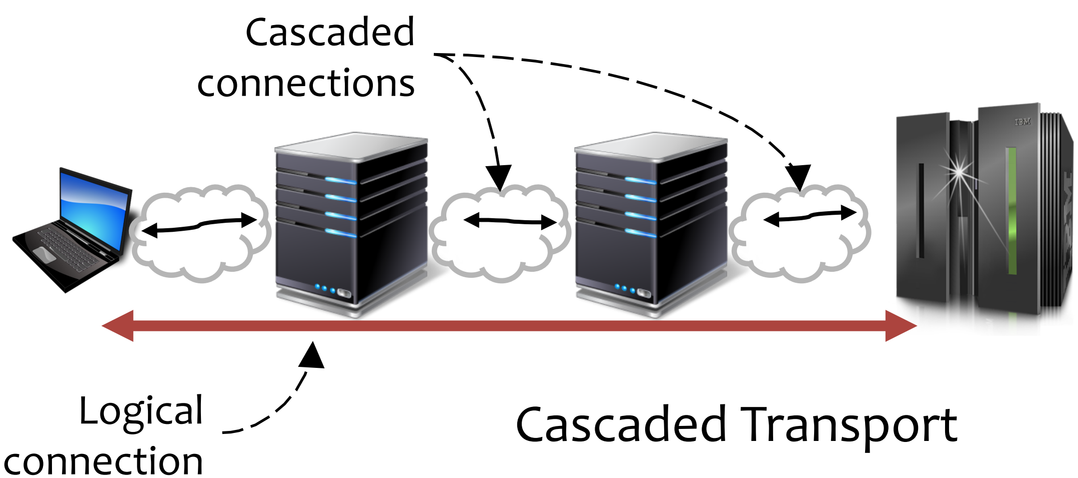
- Overloaded Abstractions
  - Coupled with roles spanning multiple layers
- No notion of:
  - Independent IDs for services
  - Session/conversation
  - Flow (/logical stream)



Research Challenges Implementation Framework?



Seamless handoff w/ WiFi calls



Cascaded Transport

- How to implement negotiation of configuration?
- What support to provide and what not to?

Research Contribution

- Backwards-compatible extensions
  - Enabling TCP extensions [PFLDNeT'10]
  - Resilient communication [ICCCN'11]
  - Virtual machine migration across networks [ICCCN'13]
- Communication model
  - Session abstraction [work in progress]
- Implementation framework
  - Case study about cascaded transport [SC'12 (Poster), GLOBECOM'13]

Other contributions

- User-space impl. of fault-tolerant transport
  - Seamless disconnection/reconnection over TCP [BBL'10]

- User-space impl. of video server migration across networks
  - Transport-agnostic extensions, using UDP [BBL'11]