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

Onus is on developers to implement new features as well as support mechanisms.

Typical TCP/IP Stack

- Application
- Transport
- Network
- Link

Modern Stack

- Application
  - Libraries
- Transport
- Network
- Link
Voice over Cellular
Voice over WiFi

Seamless handoff w/ WiFi calls
(Andriod OS)

Continue using network applications with different devices
(Apple Continuity)

Duplication of effort!

Voice & video over different networks
(Apple Facetime)

Voice over WiFi

Video

Voice

Screen cast
(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

<table>
<thead>
<tr>
<th>TCP Implementation Assumptions</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream label: socket pair</td>
<td>• Naming</td>
</tr>
<tr>
<td>Coupling of transport and network layer</td>
<td></td>
</tr>
<tr>
<td>Single interface / attachment point</td>
<td>• Flow Abstractions</td>
</tr>
<tr>
<td>Preclude multiple transports via different networks</td>
<td>• Multipath TCP</td>
</tr>
<tr>
<td></td>
<td>• Multihoming</td>
</tr>
<tr>
<td></td>
<td>• Flow Migration</td>
</tr>
<tr>
<td></td>
<td>• Hybrid Transport</td>
</tr>
<tr>
<td></td>
<td>• Session Management</td>
</tr>
<tr>
<td>Lifetime of IP address assignment greater than lifetime of the connection</td>
<td>• Mobility</td>
</tr>
<tr>
<td>Dumb network</td>
<td>• Middleboxes as 1st class citizens</td>
</tr>
<tr>
<td>Static configuration</td>
<td>• Cross layer communication</td>
</tr>
<tr>
<td></td>
<td>• Dynamic stack composition</td>
</tr>
</tbody>
</table>
Research Challenges
Communication Model

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?

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

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

- **TERMINATED**: end()
- **INITIALIZED**: create() -> end(); register()
- **AWAITING CONTACT**: await_call(); attend()
- **ESTABLISHED**: end(); reconfigure(); end(); revoke()
- **DEPARTED**: end()
- **INSTANTIATED**: join() / invite()
- **AWAITING**: add_flow()
- **REGISTERED LABEL**: await_call()
- **REVOKED LABEL**: end()

**Session Management**

**Negotiation of Configuration**

**Data Transfer**
Primitives and State Transitions

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)`

Session Management
Negotiation of Configuration
Data Transfer
Control Channel

- Enable exchange of control messages: verbs
  - Example: `refresh, suspend_flow, resume_flow`
Control Verbs

{
  "VERB": "refresh",
  "SOURCE": "alice.node",
  "TRANSACTION ID": "1872",
  "SESSION LABEL": "a.session",
  "AUTH TOKEN": "h@kl#S",
  "TIMEOUT": 20
  "PAYLOAD": ["list":
    ["list":
      ["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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Like TCP on the wire</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible w/ apps</td>
<td>✓</td>
<td>X / ✓</td>
<td>X / ✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible w/ TCP stacks</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Transport independent</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Fault tolerant</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multihoming</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Flow migration</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X / ✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>End point migration</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Session abstraction</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
## Related Work

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Like TCP on the wire</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible w/ apps</td>
<td>✓</td>
<td>X / ✓</td>
<td>X / ✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible w/ TCP stacks</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Transport independent</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Fault tolerant</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multihoming</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Flow migration</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>X / ✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>End point migration</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Session abstraction</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

Perhaps ahead of its time
## Related Work

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Like TCP on the wire</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible w/ apps</td>
<td>✓</td>
<td>✗ / ✓</td>
<td>✓ / ✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible w/ TCP stacks</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Transport independent</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Fault tolerant</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multihoming</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Flow migration</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗ / ✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>End point migration</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Session abstraction</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

Great idea; wasn’t adopted – lack of backwards compatibility
# Related Work

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Like TCP on the wire</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible w/ apps</td>
<td>✓</td>
<td>✗ / ✓</td>
<td>✗ / ✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible w/ TCP stacks</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Transport independent</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Fault tolerant</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multihoming</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Flow migration</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗ / ✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>End point migration</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Session abstraction</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

Perhaps mobile computing begins to flourish.
## Related Work

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Like TCP on the wire</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible w/ apps</td>
<td>✓</td>
<td>✗ / ✓</td>
<td>✗ / ✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible w/ TCP stacks</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Transport independent</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Fault tolerant</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multihoming</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Flow migration</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓ / ✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>End point migration</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Session abstraction</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

Multiple interfaces in mobile devices are put to use.
## Related Work

TCP is here to stay, perhaps sockets too

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Like TCP on the wire</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible w/ apps</td>
<td>✓</td>
<td>X / ✓</td>
<td>X / ✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible w/ TCP stacks</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Transport independent</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Fault tolerant</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multihoming</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Flow migration</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>X / ✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>End point migration</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Session abstraction</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

TCP is here to stay, perhaps sockets too.
## Related Work

### Lessons:
1) Avoid point solutions
2) Backwards compatibility is important

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Like TCP on the wire</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>w/ apps</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Backwards compatible</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>w/ TCP stacks</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Transport independent</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Fault tolerant</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multihoming</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Flow migration</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>End point migration</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Session abstraction</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
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

![Diagram showing logical stream over TCP]

- Sockets API
- Isolation Boundary
- Transport
- Disruption (e.g., firewall losing state, mobility)
Backwards Compatible Extensions

TCP Header

- Source Port
- Destination Port
- Sequence Number
- Acknowledgment Number
- Offset
- Reserved
- TCP Flags
  - C
  - E
  - U
  - A
  - P
  - R
  - S
  - F
- Window
- Checksum
- Urgent Pointer
- TCP Options (optional)

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

Peer_\textsubscript{A} 

Choose TIFID_\textsubscript{A} and TISeq_\textsubscript{A}

Record TIFID_\textsubscript{B}, TISeq_\textsubscript{B}

Peer_\textsubscript{B}

\begin{align*}
\text{SYN + TIFID}_\textsubscript{A} + \text{TISeq}_\textsubscript{A} \\
\text{SYN + ACK + TIAck}_\textsubscript{A} + \text{TIFID}_\textsubscript{B} + \text{TISeq}_\textsubscript{B} \\
\text{ACK + TIAck}_\textsubscript{B}
\end{align*}

Record TIFID_\textsubscript{A}, TISeq_\textsubscript{A} and choose TIFID_\textsubscript{B}, TISeq_\textsubscript{B}

Transport Independent Flow ID (TIFID)

Sequence # | 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:

Icons downloaded from iconarchive.com
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

1. Application
   - IP+port
2. Transport
   - srcIP, srcPort
dstIP, dstPort
3. Network
   - IP

Abstractions Redefined: Clean separation of concerns

1. Application
   - serviceID
2. Session
   - sessionID
3. Flow
   - flowID
4. Transport
   - transport conn.
5. Network
   - IP
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
Voice over Cellular
Voice over WiFi

Seamless handoff w/ WiFi calls (Android OS)

Continue using network applications with different devices (Apple Continuity)

Voice & video over different networks (Apple Facetime)

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

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

• Add layer below TCP?
  • Middleboxes will not let the packets through!

Backwards compatibility?
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!

Image source: software-engineer-training.com
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

Process A

Network

Process B

Transport connection
Research Challenges

- Overloaded Abstractions
  - Coupled with roles spanning multiple layers

- No notion of:
  - Independent IDs for services
  - Session/conversation
  - Flow (/logical stream)

Communication Model?

- Application
  (IP + Port)

- Transport
  (srcIP, srcPort, dstIP, dstPort)

- Network
  (IP)

- Data Link

- Physical
Research Challenges

Implementation Framework?

• How to implement negotiation of configuration?

• What support to provide and what not to?

Seamless handoff w/ WiFi calls

Cascaded Transport

Logical connection

Cascaded connections

Voice over LTE

Voice over WiFi

Icons downloaded from iconarchive.com
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]