Session Layer Resurgence

Problem situation

- Demand for new network services
- Aging Internet architecture
- Need to handle mobility and nomadicity
- Lots of extensions of TCP/IP: MIP, HIP, IPSec, ...

Proposed solution

- Adopt a more flexible view of the protocol stack
- Introduce new functionality at the session layer
- Use event-driven reconfiguration and state management
Session Layer Resurgence

Karl Palmskog

Verification of the Session Management Protocol
Session layer components

- Event collector/dispatcher
- Preferences/rules database
- Socket rebind extension
- Session API
- TCP state controller
- Session Management Protocol (SMP)
Session Layer Resurgence

- Session-enabled application
  - Session Management API
  - Session Management Protocol
  - Event collector and dispatcher
  - Rebind-enhanced socket API
    - TCP state controller
    - Preferences and rules database
    - Rebind across the stack
  - Transport layer protocols
  - Network layer protocols

- Legacy application
Session Management Protocol

- Data integrity for sessions
- Keep track of communication state
- Send and receive context updates

Background

- Developed as a part of an earlier master’s project
- Proof-of-concept implementation in the Linux kernel
- Vital part of the session layer
Problem and aim

Problem

- SMP correctness is critical
- Data integrity must be preserved
- Design must be deadlock-free

Aim

1. Understand SMP and describe it formally
2. Specify the correctness of the protocol
3. Prove that the protocol satisfies the specification
Method

Model checking

- Provide system model $M$, specification $\Phi$
- Check automatically whether $M$ satisfies $\Phi$
- Use abstraction to reduce state space

Choices

- Modelling language: PROMELA
- Specification language: Linear Temporal Logic
- Model checker: SPIN
Session Management Protocol

SMP service provisions

- Provide reliable data transfer between endpoints...
- ...despite intermittent connectivity in both space and time

SMP channels and message types

- Data channel
  - \textit{data} — application data
  - \textit{checkpoint} — communication state data
- Control channel
  - \textit{resume} — request session resumption
  - \textit{resume\_ok} — confirm session resumption
  - \textit{resume\_denied} — deny session resumption
  - \textit{suspend} — sender has suspended
Session Management Protocol

State machine

T1: Network lost
T2: User suspends; send suspend
T3: Received resume; rebind
T4: Received suspend
T5: User suspends
T6: Received resume
T7: Network changed
T8: Received resume; send resume_denied
T9: User resumes
T10: Sent resume_ok; rollback
T11: Failed to send resume_ok
T12: Sent resume
T13: Failed to send resume
T14: Received resume_ok
T15: Received resume_denied
T16: Network changed; rebind
T17: Received resume; initiator
T18: Received resume; not initiator
T19: Network lost; change interface
Verification of the checkpoint mechanism

- Lets network endpoints agree on resumeable states
- Endpoints send *checkpoint* messages when buffers fill up
- Cannot create new checkpoint until other endpoint responds
- Specification: processes always have a common checkpoint

Design flaw

- A *checkpoint* request can be interpreted as a response
- Possible to get ambiguously defined states in some situations
- Solution: only allow one endpoint to send checkpoint requests
Results

State machine correctness

- Safety: if a session is resumed, it is resumed properly
- Liveness: there are no deadlocks

State machine model

- Add control channels and states to checkpoint model
- Use Promela’s channel over channel feature for mobility
- Protocol changes due to changes in the checkpoint mechanism

Verification results

- Exhaustively verified for some parameters
- Many partial state-space searches
Conclusions and future work

Conclusions

- Produced unambiguous specification of the protocol
- Detection and correction of a design flaw
- SMP reliability has increased

Future work

- Implement changes and test them
- Verify other parts of the session layer design
- Investigate SMP/TCP interaction

“Every protocol should be considered incorrect until the opposite is proven.”
—Gerard J. Holzmann, author of Spin
Protocol models and thesis available at:
http://www.palmskog.net/exjobb

Session Layer Resurgence: Towards Mobile, Disconnection-
and Delay-tolerant Communication. Accepted to the 4th
European Conference on Universal Multiservice Networks
http://www.irit.fr/ecumn07