

Modeling Concurrent and Probabilistic Systems

Lecture 10: The Alternating-Bit Protocol

Joost-Pieter Katoen Thomas Noll

Software Modeling and Verification Group
RWTH Aachen University
noll@cs.rwth-aachen.de

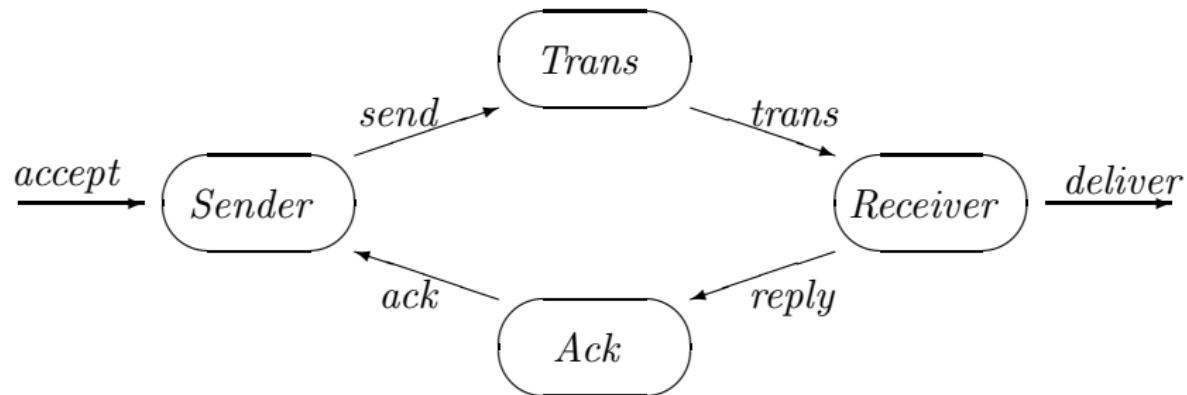
<http://www-i2.informatik.rwth-aachen.de/i2/mcps09/>

Summer Semester 2009

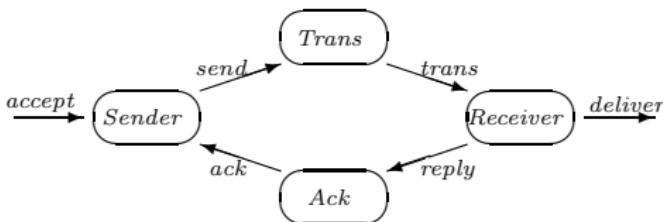
- 1 The Alternating-Bit Protocol
- 2 Implementation of the Alternating-Bit Protocol
- 3 Analysis of the Alternating-Bit Protocol

Goal: design of a communication protocol which guarantees **reliable data transfer** over **unreliable channels**

Overview of system “architecture”:



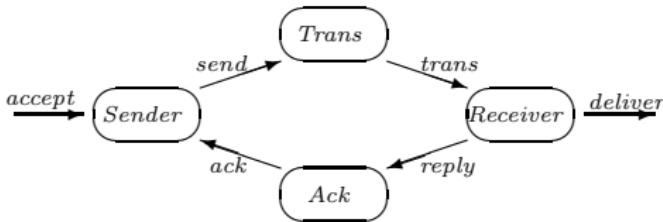
Working Principle



- *Sender transfers data* (from a given finite set D) to *Receiver* using channel *Trans*
- *Receiver confirms reception* via *Ack*
- Properties of channels:
 - **unidirectional** data transfer
 - capacity: **one message**
(\Rightarrow sequential, i.e., respects order of messages)
 - detection of **transmission errors**
(loss/duplication/corruption of messages)
 - errors **reported** to target station

Idea: use **redundancy** (additional control bit) to ensure safeness of data transfer

Modeling of Channels



- *Trans* transmits **frames** of the following form:

$$F := \{db \mid d \in D, b \in \{0, 1\}\} \quad (\text{finite})$$

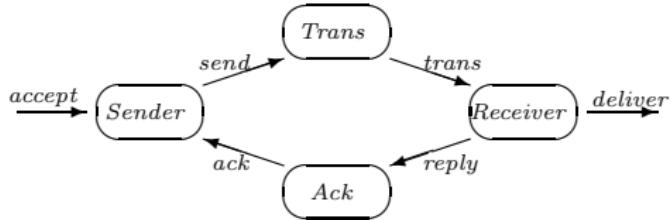
It detects **transmission errors** and reports it to *Receiver*:

$$Trans = \sum_{f \in F} send_f. (\underbrace{\overline{trans}_f. Trans}_{\text{successful}} + \underbrace{\overline{trans}_\perp. Trans}_{\text{error}})$$

- *Ack* behaves like *Trans* but transmits only **control bits**:

$$Ack = \sum_{b \in \{0, 1\}} reply_b. (\underbrace{\overline{ack}_b. Ack}_{\text{successful}} + \underbrace{\overline{ack}_\perp. Ack}_{\text{error}})$$

Design Goal



Under the above side conditions, give CCS implementations of *Sender* and *Receiver* such that the overall system **works correctly**, i.e., behaves like a **one-element buffer**:

$$Buffer(\overrightarrow{accept}, \overrightarrow{deliver}) = \sum_{d \in D} accept_d. Buffer_d(\overrightarrow{accept}, \overrightarrow{deliver})$$

$$Buffer_d(\overrightarrow{accept}, \overrightarrow{deliver}) = \overrightarrow{deliver_d}. Buffer(\overrightarrow{accept}, \overrightarrow{deliver})$$

where

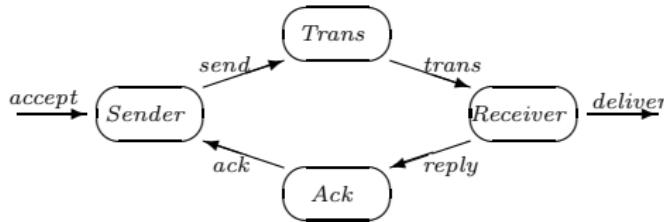
$$\overrightarrow{accept} := (accept_{d_1}, \dots, accept_{d_n})$$

$$\text{and } \overrightarrow{deliver} := (deliver_{d_1}, \dots, deliver_{d_n})$$

$$\text{for } D = \{d_1, \dots, d_n\}$$

- 1 The Alternating-Bit Protocol
- 2 Implementation of the Alternating-Bit Protocol
- 3 Analysis of the Alternating-Bit Protocol

Implementation of Sender



Sender accepts $d \in D$ via $accept_d$ and repeatedly sends frames of the form $d0$ over *Trans* until it receives the acknowledgment 0 over *Ack*. For the next data item, control bit 1 is used and so on (\Rightarrow “Alternating-Bit Protocol”).

Formally, for $b \in \{0, 1\}$ and $d \in D$:

$$\begin{aligned} Sender &= Sender_0 \\ Sender_b &= \sum_{d \in D} accept_d \cdot Send_{db} \\ Send_{db} &= \overline{send_{db}} \cdot Wait_{db} \\ Wait_{db} &= \underbrace{ack_b \cdot Sender_{1-b}}_{\text{successful}} + \underbrace{ack_{1-b} \cdot Send_{db}}_{\text{error}} + ack_{\perp} \cdot Send_{db} \end{aligned}$$

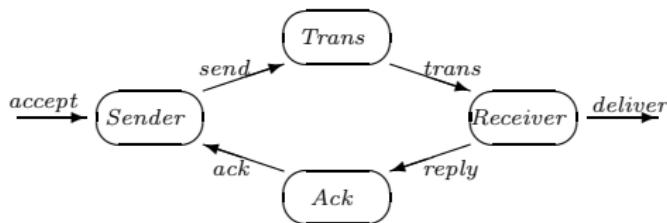
Implementation of Receiver

Receiver gets frames of the form db or \perp . In the first case, if b has the expected value, d is forwarded via $deliver_d$, and b is returned via Ack . Otherwise the transmission is re-initiated by returning the “wrong” control bit $1 - b$ to *Sender*.

Formally, for $b \in \{0, 1\}$ and $d \in D$:

$$\begin{aligned} Receiver &= Receiver_0 \\ Receiver_b &= \sum_{d \in D} trans_{db}.Reply_{db} \\ &+ \sum_{d \in D} trans_{d(1-b)}. \overline{reply_{1-b}}. Receiver_b \\ &+ trans_{\perp}. \overline{reply_{1-b}}. Receiver_b \\ Reply_{db} &= \overline{deliver_d. reply_b}. Receiver_{1-b} \end{aligned}$$

The Overall System



The **overall system** is given by

$$ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) = \text{new } L (\text{Sender} \parallel \text{Trans} \parallel \text{Ack} \parallel \text{Receiver})$$

where

$$L := \{ \text{send}_{db}, \text{trans}_{db}, \text{reply}_b, \text{ack}_b \mid db \in F \} \cup \{ \text{trans}_\perp, \text{ack}_\perp \}$$

- 1 The Alternating-Bit Protocol
- 2 Implementation of the Alternating-Bit Protocol
- 3 Analysis of the Alternating-Bit Protocol

Theorem 10.1

$$ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \simeq \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$$

Remark: because of internal τ -steps in ABP , $ABP \sim \text{Buffer}$ cannot hold.

Proof.

- ① Construct transition system of $ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$
(next slide; $S = \text{Sender}$, $W = \text{Wait}$, $T = \text{Trans}$, $A = \text{Ack}$,
 $R = \text{Receiver/Reply}$, $d, e \in D$; without restrictions)
- ② Show that $ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \approx \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$
- ③ $ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \not\sim \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$
 $\implies ABP(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}}) \simeq \text{Buffer}(\overrightarrow{\text{accept}}, \overrightarrow{\text{deliver}})$



Proof (continued).

